パソコン ライブラリ＝12

パソコンテレビX1 BASIC

戸川隼人＝著

サイエンス社

目　　次

パソコン ライブラリ＝12

BASIC

戸川隼人＝著

SHARP-HuBASIC CZ-8CB01 V1.0
Copyright (C) 1982 by SHARP/Hudson

23536 Bytes free
Ok

サイエンス社

ま　え　が　き

――――X1禮讃――――

X1はクルマに例えるならばスポーツ・カーのような魅力にあふれたパソコンである.

第一, 外観がよい. 眺めているだけでも気持のいい美しいデザインである.

また, 性能がよい. 演算が速い. 精度が高い. さらに, BASICの仕様は最高級である.

他機種にないオリジナルな機能も多い. その筆頭がテレビ画像との重ね合わせ表示である. ―― じつは筆者はその点にはあまり期待もせずに買ったのであるが, 使ってみると意外に面白い.

たとえば, ナイターを見ながらコンピューターを使えるのである. システム・テープの入力待ちや, プリンターの出力待ちの間, チャンネルをまわしてひまつぶしができる.

操作性も非常に良い. これまで使い易さで定評のあったHuBASICが標準提供されるようになったからである. ただし

(例)　**スナミ　――→　RUN**

の「カナ英字変換(?)」は惜しいことに削除されている.

MZ-80B以来シャープの伝統となっているカセット・テープの全自動制御は本機にも装備され, 一段と高速(2700ボー)になっている. 高速の自動頭出し(APSS)も可能であり, ディスク無しで十分実用になる(本機のディスクは, これまたじつにスマートですぐれたデザインなので, あればあるに越したことはないけれど).

BASICには珍しい機能がたくさんあって楽しい. 筆者が面白いと思った命令や関数は付録Cで紹介しておいた.

本書の原稿は春休みに書いた．ふだんはプロフェッサーだから，あまりパソコンにばかり熱中しているわけにいかないが，春休みは大っぴらに遊べるわけで，X1を心ゆくまで楽しむことができた．

せっかく良い機械ができたのだから，みんなで楽しく使いたい．そういう願いをこめて，この本を書いた．

みんなで広げよう．パソコンの輪．

1983年5月

H. Togawa

1　システムの説明

1.1　本　　体

シャープX1の本体CZ-800Cは，こんな形をしている．

シャープ X1 本体

この中に

CPU（中央処理装置）

メモリー（記憶装置）

インターフェース（外部の機器を接続するための窓口）

カセット・テープ・デッキ

電　源

が入っている．本機はキーボード（鍵盤）が本体と別になっている．

シャープ X1 のキーボード

1.2 CPU

本機はZ-80AというLSIをCPUとして使用している．

Z-80Aは，シャープのMZシリーズをはじめ，多くのパソコンで使われている高性能のCPUである．

Z-80A（実物大）

本機では，これをクロック周波数4MHzで使用している．これはかなり高速の使い方である．本機はさらに，BASICを実行するインタープリーターが良くできていることもあって非常に速く，8ビット・パソコンの中では最高の部類に属する．参考までに二，三の簡単な計算例の所要時間を以下に示す．

1）1から1000までの整数の和を求める（加算1000回）．

約3秒

2）1から50までの整数の乗算表（九九の表のようなもの）を作る（印刷に要する時間は除く．乗算2500回）．

全部の変数を整数型として計算した場合　　12秒
全部の変数を実数型として計算した場合　　21秒

3）$x=0^\circ$から$x=89^\circ$まで1°間隔の三角関数表を作る（sin, cos, tanを各90回計算）．

約5秒

4）30元の連立1次方程式を解く（乗算および減算それぞれ約3000回必要）．

約2分10秒

1.3 メモリー(記憶装置)

本機には64Kバイトの RAM が標準装備されている。この内，約41Kバイトは BASIC インタープリーター，モニター，入出力制御プログラムに使用されるので，ユーザーがプログラムやデーターの記憶に使用できる部分（ユーザー領域）は約23Kバイトである。

これで64Kビット，すなわち
8Kバイトの容量がある。

用　語　解　説

CPU = Central Processing Unit，中央処理装置，コンピューターの中枢部分で，演算機構および制御機構（プログラムを解読して実行するためのしくみ）を含む。

LSI = Large Scale Integration，大規模集積回路

MHz = メガ・ヘルツ，毎秒百万周期

RAM = Random Access Memory．自由に読み書きできるメモリー(要するに普通のメモリーのこと)。

ROM = Read Only Memory．読み出し専用，いいかえれば，内容を書きかえることのできないメモリー。

ビット(bit = binary digit)　2進法の1桁

バイト(byte)　コンピューターの内部でデーターを扱う際の基本的な処理単位で，8ビット（2進法の8桁）から成り，本機の場合

- 文字データーの場合，1バイトが1文字を表す。
- 数値データーの場合，5バイトで一つのデーターを表す。

KBはキロバイト（≒1000バイト = 200ワード）．**MB**はメガバイト（≒百万バイト）。

1.4 ディスプレイ

パソコンの操作をしたり計算結果を見たりするには，**ディスプレイ装置**（**モニター**ともいう）が必要である．最低限，本体とディスプレイ装置があれば，パソコンとして使うことができる．

本体の上に乗せた CZ-800D

本機に最も適したディスプレイ装置は，シャープの CZ-800D である（写真上）．これは次のような特長がある．

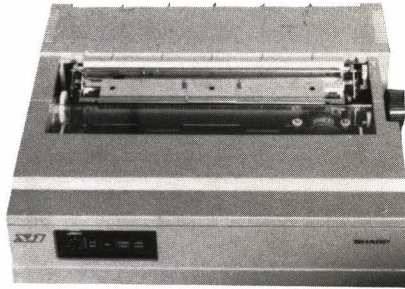
本機の出力用としてもテレビとしても使用できる
本機の出力をテレビ画面と同時に（重ね合わせて）表示できる
本機でテレビのチャンネル，音量等をコントロールできる
画面は精細，色も鮮かで美しい
外観，寸法ともに本機に合わせてある

本機を，いわゆる「パソコン・テレビ」として使うには，これを用いる必要がある．

しかし上記の諸点にこだわらなければ，マイコン・ショップの店頭に並んでいる各種のディスプレイ装置（モノクロまたは RGB 分離方式のカラー・ディスプレイ）を使用できる．また RF コンバーターを介して家庭用テレビに接続してもよい．

1.5 プリンター

パソコンの出力(計結果など)をディスプレイに表示するだけで用の済むこともあるが，紙に印字したい場合も少なくないであろう．そのためのプリンターを付けることができる．

本機のためのプリンターとしては，シャープから専用機CZ-800Pが発売されている．

CZ-800P

これを用いれば，画面コピー等が自由にできて便利である．

本機のプリンター・インターフェースは，いわゆるセントロニクス仕様準処という方式（他の大部分のパソコンと同じ）なのでCZ-800P以外のプリンターをつないでも動くはずであるが，本機はプリンター接続用のコネクターが標準のものと異なり，特殊な小型コネクターを用いているので，接続用ケーブルを自作(またはマイコン・ショップに頼んで作ってもらう)する必要がある．

[CZ-800Pの仕様]

印字速度	95字/秒	(非常に速い)
1行の字数	80字(標準)	96字（縮小文字の場合）

[インターフェース]　本機はプリンター・インターフェースを内蔵(標準装備)している．ケーブルはプリンターに付いていくる．

1.6 カセット・テープ

本機はカセット・テープの記録，再生装置を内蔵している．これは全自動(BASICのプログラムで起動，停止はもちろんのこと，早送りや巻き戻しも可能)，高速（2700ボー）で，信頼性も高い．

テープ　テープは普通の音楽用カセット・テープでもよいが，最近はパソコン用のカセット・テープを容易に入手できるようになったので，なるべくそれを使う方がよい（シャープ，ソニー，マクセル等の製品があり，「コンピューター用」といっても値段は高くない）．プログラムの保存用にはC-10（片面5分，両面10分）で十分である．

シャープのC-15

1.7 その他

以上のほか，

フロッピー・ディスク*（5インチ×2）	CZ-800F
デジタル・テロッパー	CZ-8DT
ジョイスティック	（2個）
外部RAM*	
漢字ROM*	

などを接続することができる。

---

* 拡張I/Oポート経由

2 操 作 法

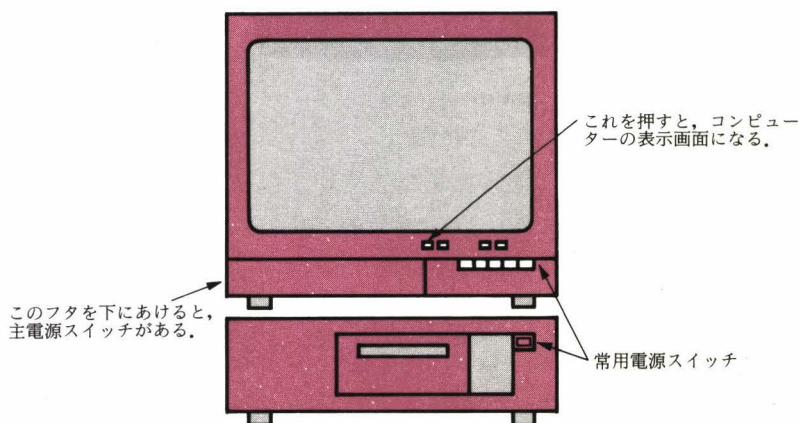
2.1 電 源 投 入

本体および本機専用ディスプレイ(CZ-800D) は，テレビの自動オン・オフを行なう関係上，電源が

主電源スイッチ　　—— 普通，入れっぱなしにしておく
常用電源スイッチ　—— 使用するときだけ入れる

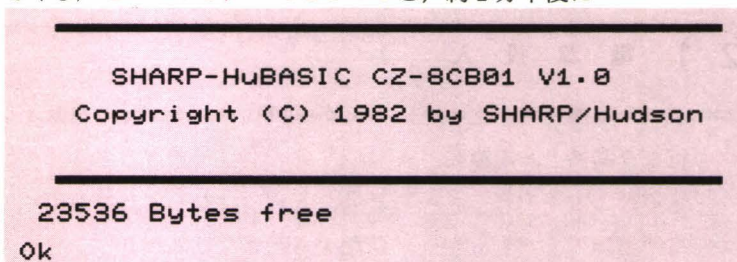
の二段式になっている．主電源を入れておかないとタイマーが正常に働かない．

本体の主電源スイッチは後面（後から見て左端）
本体の常用電源スイッチは前面右上
CZ-800D の主電源スイッチは前面左下のふたの中
CZ-800D の常用電源スイッチは前面右下

にある．

2.2 BASICの起動

電源を入れると，間もなく自動的にカセット・テープ装置のふたがあくから，BASICのシステム・テープCZ-8CB01（本体の付属品として付いてくる）をセットし，ふたをしめると，約2分半後に

```
SHARP-HuBASIC CZ-8CB01 V1.0
Copyright (C) 1982 by SHARP/Hudson

23536 Bytes free
Ok
```

というメッセージが出てBASICを使用できる状態になる．

システム・テープは直ちに自動的に巻き戻されるので，それが止まったら EJECT ボタンを押して取り出しておく．

BASICのシステム・テープ

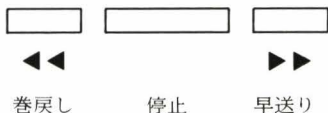
［備考］ 電源を入れたとき，既にカセット・テープが入っていれば，直ちにそのテープの読み込みが始まる．したがって，あらかじめBASICのシステム・テープを入れておけば，自動的に読み込まれ，BASICを使用できる状態になる．ただし，本機のカセット・テープ関係のメカニズムは電磁制御になっているので，電源を入れる前に EJECT ボタンを押しても，ふたはあかないわけで，いま述べたような全自動スタートをやりたければ，前回の実行時の最後にシステム・テープを入れておく必要がある．

2.3 カセット・テープからの読み込み

新しいプログラムをキーボードから入れる方法は，次節から詳しく説明するが，とりあえず既存のプログラム（友人の作品，市販のゲーム，本体の附属品として入っているデモンストレーション用プログラム等）を読ませて実行してみようというのであれば，次のようにすればよい．

1）EJECT ボタンを押すとふたがあく．

2）テープを入れて，ふたをしめる．

3）巻き戻す*．巻戻しボタンはキーボードの右上にある．

4）読み込み開始を指令する．それには，キーボードから

LŌAD "プログラムの名前" ↵

と入れるのが正式であるが，略して単に

LŌAD ↵

としてもよく（その場合，最初に見つかったプログラムが読み込まれる），あるいは，SHIFT キーを押しながらファンクション・キー（キーボードの左上段にある）F1 を押してもよい．

5）まちがったテープをかけてしまって，読み込みの途中で中止したい場合は，SHIFT キーを押しながら BREAK キーを押す（停止ボタンで止めてもよいが，それだとテープは止まっても BASIC の「読み込み状態」は解除されない）．

---

* 要するに，入力したいプログラムが，テープの「これから読む部分」に入っていればよい．なお本機では，前方，後方（各1プログラム分）の「自動頭出し」が可能である．それには，SHIFT キーを押しながら，前方なら ▶▶，後方なら ◀◀ キーを押せばよい．

2.4 キーボード

キーボード(鍵盤)には

文字入力用のキー

数字入力用のテン・キー

操作用のキー

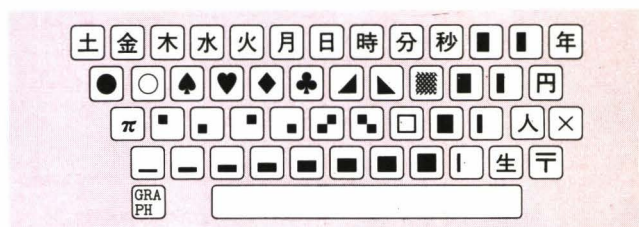
が並んでいる.

[大文字の入力]　BASICのプログラムを入力するときは，左下にある CAPS LOCK (capital letters, すなわち大文字の状態にロック——固定する，という意味）というキーを押し下げておくとよい．そのようにすると，キーの頭に大きく刻印されている文字（下図参照）が入力される.

[小文字と記号の入力]　SHIFT キーを同時に押すと次のような文字が入力される.

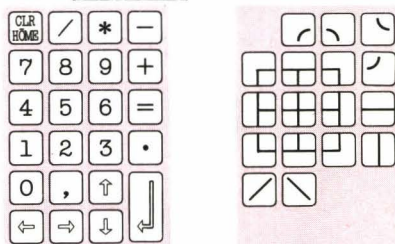
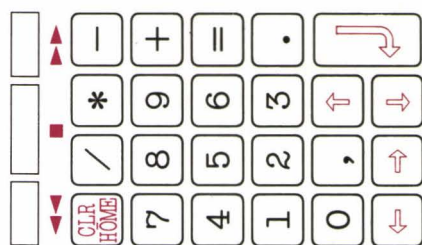
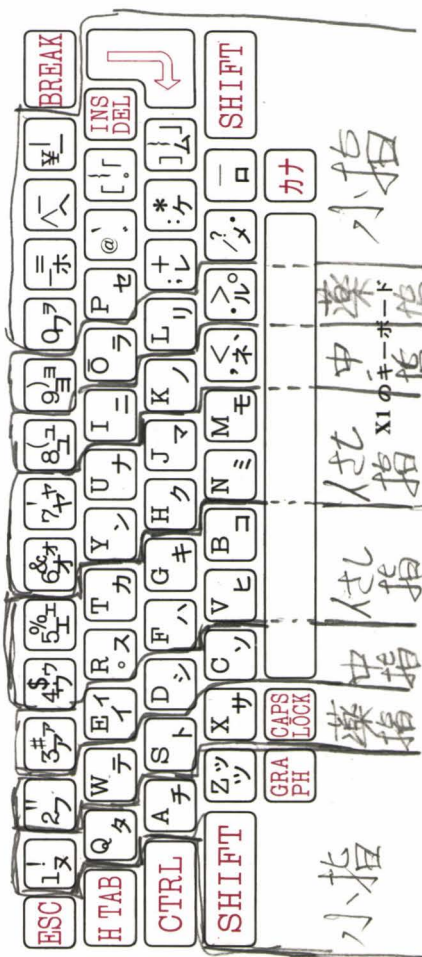
カナ というキーを押しておくと（ロックされる）次のようになる.

カナ文字入力の状態でさらに SHIFT キーも押すと次のようになる.

GRAPH というキーと同時に押すと次のようなグラフィック・シンボルが入力される.

テン・キーの配列は次のようになっている.

（左は普通の場合，右は GRAPH キーを押した場合）

F1
F2
F3
F4
F5
∨CHANNEL∧
∨VOLUME∧
COMPUTER/TV
ESC
H TAB
CTRL
SHIFT
GRAPH
CAPS LOCK
カナ
INS DEL
BREAK
SHIFT
CLR HOME

X1 のキーボード

2.5 画 面 操 作

画面消去　SHIFT キーを押しながら CLR/HŌME というキー（テン・キーの左上にある）を押すと，画面に表示されている文字は全部消えて，カーソルは左上隅に移る．

図形（グラフィック出力）は，上記の操作では消えない．図形を消したいときは，キーボードから次のように指令する．

CLS 0 ⏎

[備考]　**CLR** は clear（クリアー），**CLS** は clear screen（画面クリアー）の意味である．

1行の字数の指定　電源を入れた直後には，1行に40字を表示する状態になっているが，

WIDTH 80 ⏎

と指令すれば，1行に80字の表示をすることができる．これをまた40字/行に戻すには

WIDTH 40 ⏎

とすればよい．なお，画面の行数は25行で，これは変更できない．

色の指定　カラー・ディスプレイを使用しているのであれば，気分転換のため（？）に文字の色や背景の色を変えることができる．

CŌLŌR 文字の色，背景の色

で指定する．色は下記の色番号で表す．

0	1	2	3	4	5	6	7
黒	青	赤	紫	緑	水色	黄	白

2.6 プログラムの入力

NEW　新規にプログラムを入力するときには，まず

NEW ⏎

を入れる．

[解説] これは「新しいプログラムを入れますよ」という，パソコンに対する意思表示である．これを入れると，パソコンは，それまで入っていたプログラム（たとえば，前に使った人が残していったもの）を消して，新規入力のための準備を行なう．ただし，電源を入れて最初に使うときは，**NEW**を入れなくてもよい．

基本的な入力方法　BASICのプログラムやデーターは，行単位に入力する．各行の終りには，必ず ⏎ キーを押す．

(例)

```
10  A=2 ⏎
20  B=3 ⏎
30  C=A+B ⏎
40  PRINT  C ⏎
RUN ⏎
```

このプログラムの意味は3.2節で説明する．しかし，とりあえず，左記のとおりに入力してみるとよい．2+3の値5が表示されるであろう．

[解説] 上の例のように行番号を付けて入力すると，プログラムとして記憶装置に格納され，あとで**RUN**（実行せよ）という指令が出たとき（原則として行番号の順に）実行される．

それに対し，行番号を付けないで入力すると「直ちに実行する命令」（これを直接実行形式という）として解釈され，入力時点ですぐ実行され，記憶装置には残らない．

命令 { 行番号付き…………プログラムとして記憶される．
　　　 行番号なし…………直ちに実行される．

2.7 実行の開始，停止，再開

開　始　プログラムの実行を開始するには，キーボードから

RUN ⏎　または　R. ⏎　または　F5

を入れればよい．

RUN 開始行番号 ⏎ という使い方もできる．

停　止　プログラムの実行を止めるには，SHIFT キーを押しながら BREAK キーを押せばよい．

他社のパソコンだと，たいてい BAEAK キーまたは STŌP キーを押せば止まるが，シャープのパソコンは SHIFT キーをいっしょに押さないと止まらない(MZ シリーズに共通)．うっかりさわったために止まってしまうのを防止するためと思われる．

再　開　実行を再開するには次のようにする．

CŌNT ⏎　または　C. ⏎

CŌNT は continue（続ける，の命令形）の意味である．

標準状態にもどす　図形表示関係のプログラムや操作法に誤りがあると，カーソルが出なくなったり，点滅が止まらなくなったりして困ることがある（これは SHIFT + BREAK では解決されない）．そういう場合，

CTRL + D

すなわち CTRL キーを押しながら D のキーを押せば標準状態にもどり，制御機能を回復することができる．

これは本機独特の便利な機能である．

2.8 行番号の自動生成

BASICのプログラムを入力する際には各行に行番号を付けなければならない．それをいちいちキーボードから入れるのは手間がかかるので，自動的に行番号を付ける機能が用意されている．それには**AUTŌ**というコマンド（コンピューターに対する指令）を用いればよい．

AUTO　　1）最も簡単な形

AUTŌ [↵]　　または　　[F1]

このように指令すると，

10，　20，　30，　40，　50，　…

というように，10から始まる10間隔の番号が生成される．

2）最初の番号を指定するには

AUTŌ　最初の行番号 [↵]

とすればよい．

(例)　AUTŌ 800 [↵]

3）間隔も指定するには

AUTŌ　最初の行番号，間隔 [↵]

とする．間隔は正の整数でなければいけない．

(例)　AUTŌ 300,1 [↵]

[使用法]　**AUTŌ**を指令すると，画面の左端に行番号が表示されるから，その右に文をキーボードから入れて[↵]キーを押せばよい．

[重複防止]　自動生成した番号が，既に使われている（既に入力したプログラムと重複する）場合には，現在の内容が表示されるから，そこで**AUTŌ**を解除すればよい．

AUTOを解除する方法　　自動番号付けを止めるには[SHIFT]キーを押しながら[BREAK]キーを押す．

2.9 プログラムの清書と番号整理

LIST　プログラムを入れ終ったら

LIST ⏎　または　L. ⏎　または　F4

によって，いま入力したプログラムを表示させ，正しく入力されているかどうかを調べてみるとよい．

プリンターに出力したい場合は，LLIST ⏎ とする．

[表示の中断]　長いプログラムで1画面に表示できない場合，**スクロール**といって，表示内容を1行ずつ上にずらして，一番下の行をあけて，そこに新しい行を表示する．そのため，画面の上の方に表示されている行は，上から1行ずつ画面の外に追い出されてしまう．したがって，途中で停止させないと，ゆっくり画面を見ることができない．LISTを中断するには BREAK キーを押せばよい．(本機ではこの目的に ESC キーを使うことはできない．)

[表示の再開]　表示を中断したあと，続きを見るために表示を再開するには，スペース・バーを押す．

RENUM　挿入や削除によって不等間隔になった行番号を

RENUM ⏎　または　REN. ⏎

によって，きれいに付け直すことができる．後述のGŌTŌ文などの行先は，それに合わせて自動的に修正される．

RENUM　最初の番号 ⏎

によって，付け直したあとの番号の出発値を指定することもできる．

RENUN　新行番号，旧行番号，間隔

という形式で指定すれば，「旧行番号」から先の部分だけを，新行番号から始まる指定間隔の番号に付け替えることができる．

2.10 誤字訂正

プログラムの入力中に誤りに気付いたときは，次のようにして誤りを訂正することができる．

> 1）行の途中（[⏎]キーを押す前）で誤りに気付いた場合は，カーソルを誤りの所までもどして正しい文字に直せばよい．

（例）　10　A=1　を誤って，10　A−1　と入力し，[⏎]を押す前に気付いた場合，[←]キーを2回押して，カーソルを−の位置に合わせ，＝のキーを押せば

```
10 A=1
```

になるから，そこで[⏎]キーを押せばよい（[⏎]を押すとき，カーソルは，この行の中のどの位置にあってもよい）．

> 2）[⏎]キーを押したあとでも，まだ誤りの箇所が画面に表示されていれば，カーソルをそこにもどして正しい文字に直し[⏎]キーを押す．

（例）

```
10 S=2
20 B=3
30 C=A+B
40 ■
```

（■印はカーソル）

まで入力した時点で，最初の行の「S」は「A」の誤り，ということに気が付いたときは，[↑]を3回押せば

```
10 S=2
```

になるから，ここで[A]を押し，[⏎]キーを押し，[↓]と[→]で先ほどの位置までカーソルをもどして，続きの部分を入力すればよい．

> [注意]　AUTOで自動的に行番号を付けている場合は，[SHIFT]＋[BREAK]によりAUTOを解除してから上記の操作を行うこと．

3）誤りの箇所が画面上にないときは，

EDIT　訂正したい行の番号 ⏎

によって，誤りの部分を画面に表示し，そこにカーソルを合わせて訂正し ⏎ キーを押す．

[**備考**] EDITのかわりにLISTを用いてもよい．

(例) 行番号30を誤って

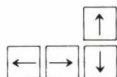
30 C=A；B

と入力していたことに，あとで気付いた場合は，

LIST 30 ⏎

と入れると，画面に表示されるから，カーソルを；の位置に合わせて修正する．

カーソル　カーソル（cursor）とは画面上で現在の入力可能な位置を指している印のこと．点滅している．

カーソルの動かし方　テン・キーの下方にカーソルを移動するためのキーがあり，移動方向が矢印で示されている．

訂正しないでよいもの　後述の文字定数などの特殊な場合以外は

スペースの入れ過ぎ，入れ忘れ

大文字と小文字の混同

は無害であるから訂正しないでもよい．

[**注意**] 画面上で訂正を行なったあと，必ず ⏎ を押すこと．⏎ キーを押さないと，画面上では直っていても，メモリーの内容は訂正されない．

2.11 挿　　入

文字の挿入　画面に表示されている行の一部に文字を挿入するには，次のようにする．

1）挿入したい位置にカーソルを合わせる．

2）SHIFT キーを挿しながら INS/DEL* キーを挿入したい文字の数だけ押す．

3）挿入したい文字のキーを押す．

（例）画面の表示が

30 CA+B

となっていて，**C**と**A**の間に＝記号を挿入したい場合，**A**の所にカーソルを合わせて

30 CA+B　　　　（■印はカーソル）

として，SHIFT キーを押しながら INS/DEL キーを1回押すと，

30 C A+B

になるから，＝のキーを押し，⏎ を押す．

[別の方法]　FM-8 や PC-8801 と同じような「挿入モード」方式による修正も可能である（23ページ CTRL + A 参照）．

行の挿入　BASIC のプログラムは各行に行番号が付いている．新しい行を挿入する必要が起こったら，既存の行の中間の行番号を付けて入力すればよい．

（例）行10と行20の間に**B=3**を挿入したい場合，10と20の間の適当な数（たとえば**15**）を行番号として

15 B=3 ⏎

とすればよい．

---

* **INS**は insert（挿入），**DEL**は delete（削除）の略．

2.12 削　　除

削　除　入力した文字を削除するには，普通，次のようにする．

削除すべき位置より1字右にカーソルを合わせる

INS/DEL キーを押す

(例)　まちがえて

30 PURINT C

と入力し，文字Uを削除した場合，その1文字右の位置だからRの所にカーソルを合わせて

30 PURINT C

とし，INS/DEL キーを押せば

30 PRINT C

となる（削除の場合のカーソルは「左向きに移動する文字列の先頭車」というつもりで用いるとよい）．

行の削除　削除したい行の行番号に続けて ⏎ キーを押す．

(例)　行100を消すには次のようにする．

100 ⏎

何行も続けて消したい場合は**DELETE**を用いるとよい．削除すべき範囲は次のように指示する．

DELETE　先頭の行番号－最後の行番号

または

DELETE　先頭の行番号，最後の行番号

(例)　DELETE 30－80

指定番号から先を全部消したいときは

DELETE　先頭の行番号－

指定番号までを全部消したいときは

DELETE　－最後の行番号

とする．綴り**DELETE**は　D．　で代用できる．

2.13 コントロール・キー

（初心者はここをとばしてよい．）

コントロール・キー　キーボードには CTRL というキーがある．これはコントロール・キーといって，単独に押したのでは何の効果もないが，これを文字のキーと同時に押すことにより，下記のような各種の画面操作を行なうことができる．

CTRL + E　**右側消去**　現在のカーソル位置から，その行の右端（文法上の1行が画面上で二つ以上の行に分割して表示されている場合は文法上の行の終り）までを消去．

CTRL + Z　**下側消去**　現在のカーソル位置より下の行を全部消去．

CTRL + N　**上向き行あけ**　現在のカーソル位置より上の部分を上向きにスクロールして行のすきまを作る（行の挿入用）．

CTRL + O　**下向き行あけ**　現在のカーソル位置より下の部分を下向きにスクロールして行のすきまを作る（行の挿入用）．

CTRL + J　**行の分割**　現在のカーソル位置の直前で行を二つに分ける．たとえば ⏎ を入れ忘れて

```
50 A=B 60 PRINT C
```

と入力してしまった場合，6の所にカーソルを合わせて CTRL と J を押せば

```
50 A=B
60 PRINT C
```

となり，ここで ⏎ を押せば，行50，行60ともに上の形に修正される．

CTRL + W　**行の連結**　現在カーソルがある行とその次の行をつないで一つの行にする．ただし余分な空白をつめてくるるわけではないので，画面は変化しない．「次の行」に行番号が付いている場合，警告のために#印が付く．

CTRL＋A　**挿入モードの開始，終了**　INS/DEL キーによる挿入は，字数が少ないときはべつに問題ないが，字数が多い場合には，いちいち字数を数えるのがめんどうである．そういう場合，挿入したい位置のすぐ右にカーソルを置いて CTRL＋A とすると，画面は変らないが「挿入モード」になり，挿入したい文字のキーを押すと，その字数分だけ自動的にすきまが作られて（カーソル位置から右の部分が右に移動し）キーボードから入れた文字が挿入される．

（例）　画面の表示が

```
30 CA+B
```

となっていて，CとAの間に＝記号を挿入したい場合，Aの所にカーソルを合わせて CTRL＋A を押し，＝を入れればよい．

[解除方法]　挿入モードから抜け出すには，再度 CTRL＋A とすればよい．なお，それをしなくても，カーソル移動キーまたは ⏎ キーを押せば解除される．

CTRL＋F　**1語スキップ**　次の単語* の先頭位置にカーソルを進める．

（例）

```
30 MENSEKI=TEIHEN*TAKASA/2
30 MENSEKI=TEIHEN*TAKASA/2
30 MENSEKI=TEIHEN*TAKASA/2
30 MENSEKI=TEIHEN*TAKASA/2
30 MENSEKI=TEIHEN*TAKASA/2
```

CTRL＋B　**1語バック**　今の単語* の先頭位置までカーソルをもどす．ただしカーソルの現在位置が単語の先端にあるときは，一つ前の単語の先頭位置に移る．

---

*　ここで単語というのは英数字の列，いいかえれば「記号または空白の直後から，記号または空白の直前まで」のこと．

2.14 TAB機能

(初心者はここをとばしてよい。)

タイプライターに TAB というキーがある. tabulate すなわち作表のための機能で，このキーを押すと，あらかじめ指定してあった文字位置までスキップするので，各行の項目の頭をそろえるのに便利である．本機にもそれと同じ機能があり，キーボードの左端にある HTAB* というキーを押すと，カーソルは現在位置の次(右)のTAB位置に移動する.

特に指定しなければ，TAB位置は左端を起点として8字間隔に設定されている.

(例)

画面左端

12345678123456781234567812
ABCDEFGHIJKLMNŌPQRSTUVWXYZ

自分の好きな位置に「TABの行先」を設定したり解除したりするには，その位置にカーソルを置いて，次のようにすればよい.

設定の場合 CTRL + T

解除の場合 CTRL + Y

(例) BASICのプログラムは普通は左につめて入力するが，本書のプログラム例（たとえば74ページ参照）のように段落を付けておくと見易く，誤りも少なくなってよい. このような段落を付けるのに，いちいちスペース・バーを押すのでは大変だから，

の＊印の所にTABをセットしておいて，HTAB を用いるとよい.

＊ HTABは horizontal TAB の略.

2.15 ファンクション・キー

キーボードの左上段に

F1　F2　F3　F4　F5

[　] [　] [　] [　] [　]

というキーが並んでいる．これはファンクション・キーといって，

簡易操作用

割り込み用

の二つの使い途がある．この内，簡易操作用としては，次のような機能をもつ*．

[F1]	行番号自動生成（AUTŌ [⏎] と同じ）
[F2]	現在時刻を表示（?TIME$ [⏎] と同じ）
[F3]	KEY**
[F4]	画面の下側を消してプログラムを表示
[F5]	実行開始（RUN [⏎] と同じ）
[SHIFT]+[F1]	プログラムの読み込み開始（LŌAD [⏎]）
[SHIFT]+[F2]	WIDTH**
[SHIFT]+[F3]	CHR$(**
[SHIFT]+[F4]	PALET**
[SHIFT]+[F5]	実行再開（CŌNT [⏎] と同じ）

[注意]　[F1], [F2], [F4], [F5], [SHIFT]+[F1], [SHIFT]+[F5] を用いる場合は，あらかじめカーソルを「空いている行」に置いてからファンクション・キーを押すこと．

---

*　標準状態，すなわち後述のKEY文による機能変更をしていない場合．

**　これらの綴りが表示される．この後にパラメーターを補って命令あるいは関数にして用いる．

2.16 カセット・テープへの記録

できあがったプログラムをカセット・テープにセーブ（save, 保存のため書い込むこと）しておいて，後日それをロード(load, 読み込んで制御機構に装着すること）することができる．

現在メモリーに入っているプログラムをカセット・テープに記録するには，正式には

SAVE "CAS：ファイル名" ⏎

とする．しかし普通は**CAS：**を省略して

SAVE "ファイル名" ⏎

としてよい．このようにしてファイル名を付けておけば，1本のテープに二つ以上のプログラムを記録しても，指定したファイルを自動的にさがして読み込むことができて便利である．

ファイル名を付ける必要がなければ，**SAVE** ⏎ だけでよい．1本のテープにプログラムを1本しか記録しない場合にはこれでもよい．また，ファイル名を使うのがめんどうだからといって全部のファイルに同じ名前を付ける人がいるが，そうするくらいならファイル名を省略する方が簡単でよい．

ファイル名の付け方　本機では13字までのファイル名を用いることができる．英字，数字，カナ文字，空白，記号（ただしコンマとコロンとセミコロンを除く）を使用してよい．

照　合　プログラムをセーブしたとき，カセット・テープに正しく記録されているかどうか調べるには，セーブしたあと，テープを巻き戻し，次のように指令する．

LOAD? "ファイル名" ⏎

3　BASIC 入門——簡単な計算と入出力

3.1　BASIC とは

BASIC は，コンピューターを使うための言語（プログラミング言語）の一種である．ほかにも

FORTRAN　　COBOL　　PL/I　　PASCAL

など各種のプログラミング言語があるが，BASIC には

簡単で覚え易い

という特長があり，そのためパソコンでは広く使われている．

BASIC は1964年，Dartmouth 大学の J. G. Kemeny と T. E. Kurtz により，学生にコンピューターを教えるための言語として開発され，教育用言語として，また TSS 用言語として知られていたが，パソコンで使われるようになってから機能は大幅に拡張され，今日では実務にも広く用いられている．一口に BASIC といってもいろいろな種類がある．本機の BASIC（SHARP HuBASIC という）は，

整数型，実数型，倍精度実数型を使用できる

GŌTŌ 等の行先にラベルを使用できる

配列は何次元でもよい（文字列型の多次元配列も可）

IF～THEN～ELSE が使える

文字列の処理機能が強力

図形表示機能が強力

テレビの制御や重ね合わせ表示ができる

スクリーン・エディターが使い易い

などの特長をもつ，たいへんデラックスな BASIC である．

3.2　命令とプログラム

直接実行形式　　パソコンは命令どおりに動く．たとえば

　　A=2　　　　（**A**の値を**2**にせよ，という意味）

と命令すると，変数**A**の記憶場所を用意し，そこに値**2**を入れる．また

　　B=3　　　　（**B**の値を**3**にせよ，という意味）

と命令すると，変数**B**の記憶場所を用意し，そこに値**3**を入れる．次に

　　C=A+B　　　（**A+B**の値を**C**に代入せよ）

と命令すると，変数**C**の記憶場所を用意し，**A+B**すなわち**2+3**の計算をし，結果**5**を**C**の所に入れる．さらに

　　PRINT C　　　　（**C**の値を表示せよ，という意味）

と命令すると，**C**の値が表示される．

［実習］　もし手もとに本機があったら，上の例を実際にやってみるとよい．すなわち

　　A=2 [↵]
　　B=3 [↵]
　　C=A+B [↵]
　　PRINT C [↵]

と入力してみるとよい．即座に結果**5**が次の行に表示されるであろう．なお，ついでに

　　PRINT A,B [↵]　　　（**A**と**B**の値を表示せよ）

と命令すれば，**A**と**B**の値が確かに**2**と**3**になっていることを確認することができる．

```
A=2
OK.
B=3
OK.
C=A+B
OK.
PRINT C
 5
OK.
PRINT A,B
 2             3
OK.
```

間接実行形式（普通の使い方）　左記の例のように，1ステップごとに使用者が命令する方式では，複雑な計算はできない．そこで普通は一連の命令（これをプログラムという）をあらかじめコンピューターに記憶させ，命令を順にとり出してきて実行する，という方式（プログラム記憶方式，間接実行形式）を用いる．

BASICでこれを行なうには，命令の頭に行番号を付けて入力し，プログラムを入れ終ったら

RUN ⏎

というコマンド（直接実行命令の一種）で実行を指令すればよい．

行番号　行番号としては1～65534の範囲内の整数値を用いることができる（0は使えないので注意！）．

プログラムは原則として行番号の順に実行される．行番号はまた，プログラムの修正，挿入，削除の際の，位置の指定に用いられる．行番号は連続した番号でなくてもよい(欠番があってもよい)．そこで普通は

10，　20，　30，　40，　50，　…

というように10とびの番号を付ける*．

[実習]　左記の例を間接実行形式で実行させてみよう．それには

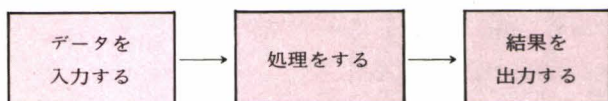
```
10  A=2 ⏎
20  B=3 ⏎
30  C=A+B ⏎
40  PRINT  C ⏎
RUN ⏎
```

と入力すればよい．行40を入れたときには何も出力されず，**RUN**させてはじめて結果**5**が表示されるであろう．

---

*　AUTŌというコマンドで「自動番号付加」を指令しておくと，自動的にこのような番号を付けてくれる（16ページ参照）．

3.3　INPUT 文

プログラムは，普通，

という構造をしている．本章では，これに従って

入力の部分の書き方

処理の部分の書き方

出力の部分の書き方

の順に文法を説明することにする．

INPUT 文の基本形　データ入力のためには INPUT という命令がある．最も単純な書き方は

```
        INPUT  変数名
(例)    INPUT  A
```

で，その変数の値を入力せよ，という意味になる．二つ以上の変数の値を一つの INPUT 文で入力するときは，変数名をコンマで区切って書く．

```
        INPUT  変数名，変数名
        INPUT  変数名，変数名，変数名          etc.
(例)    INPUT  A，B，C
```

入力要求メッセージの出し方　書き方は次のとおり．

```
        INPUT  "メッセージ"；変数名，…，変数名
または
        INPUT  "メッセージ"，変数名，…，変数名
```

上のように"メッセージ"の次をセミコロンにすると？印が表示される．それに対しコンマにすると？印は表示されない．

綴り　INPUT　は　I．　で代用できる．

(例) **INPUT "A,B=";A,B**

INPUT "xヲイレテクダサイ";X

INPUT "アナタハ ナンサイデスカ";N

入力要求メッセージは，実行時に画面に表示される．たとえば，上の最初の例の**INPUT**文を実行すると，

A,B=?

と表示され，その右にカーソルが来る．

実行時の操作方法

1) 入れるデータが1箇の場合は，その値をキーボードから入れて最後に[↵]キーを押す．

2) 二つ以上のデータを入れる場合はコンマで区切る．一つの**INPUT**文に対する入力値は，必ず1行で入れなければいけない*．

3) **INPUT**文に書かれた変数の箇数（mとする）と，入力したデータの箇数（nとする）が合わない場合**，

$m > n$ ならば，残りの変数の値は変らない．

$m < n$ ならば，余分なデーターは無視される．

指数部の付け方　非常に大きい数や非常に小さい数を入力する場合には，指数部付きの形で表すと便利である．書き方は

±整数部．小数部**E**±指数部
↑符号　　　↑指数部の符号

で，指数部**E**±nは「$\times 10^{\pm n}$」を表す．

(例) **−1E6** は $-1\times10^{6} = -1000000$

3.2E−1 は $3.2\times10^{-1} = 0.32$

を表す．

---

* 画面では何行かにまたがってもよい．画面の方は右端に来ると自動的に次の行に移ってくれるから，かまわず続けて入力する．文法上は「[↵]を入れるまでが1行」として扱われる．

** こういう場合の扱い方は機種によって異なるので，他機種のプログラムを本機に移植する場合には十分注意されたい．

3.4 代入文 (LET 文)

代入文の正式の書き方は

LET　変数名＝式

である．しかし　LETを省くことが許されているので

変数名＝式

という形で書くのが普通である．

（例）　LET　C=A+B

C=A+B

これらの文は，「＝の右に書かれた式の値を計算して，＝の左に書かれている変数に代入せよ」という意味になる．

（例）　Aの値が2，Bの値が3のとき，代入文

C=A+B

実行すればCの値は5になる．

[備考]　1）代入先の変数名は必ず＝の左側に，式は必ず＝の右側に書かなければいけない．したがって，次のような書き方は許されない．

LET　A+B=C

2）単独の変数または定数も，広い意味での「式」の一種とみなされる．したがって

LET　A=1

などと書いてもよい（「1」だけでも「式」である）．

3）変数の値は代入文によって何度でも書きかえることができる．その特別な場合として，左辺の変数名が右辺の式に現れてもよい．その場合，式の値を計算する段階では代入前の値が使われ，その結果が左辺の変数の新しい値として代入される（書きかえられる）．

（例）　Iの値が2のとき，代入文

I=I+1

を実行すると，まずI+1（すなわち2+1）の値が計算され，その結果3がIに代入されるから，実行後のIの値は3になる．要するに，この代入文は「Iの値を一つふやせ」という命令だと思えばよい．

3.5 変 数 名

本機の変数名の名付け方の規則は次のとおり．

　最初の文字は英字に限る

　2字目以後は英字または数字

　字数は240字以下

　小文字も使用できるが大文字と同じ文字と解釈される

　予約語（BASICで特別な意味をもつ綴り，たとえば**RUN**，**INPUT**など）で始まるものはいけない

　FNで始まる綴りは使用できない

なお

　識別には綴りの最後まで全部が有効である

（変数名として使える綴りの例）

　SHARP　X1　JA1AC　H2Ō　JBL

　DIFFERENCE　YMŌ　SEŌHAYAMI

（変数名として使えない綴りの例）

ŌNDŌ	予約語**ŌN**で始まる
TANKA	予約語**TAN**で始まる
H.T	英字，数字以外の文字を含む

［解説］　他機種では「識別には先頭の2文字だけが有効」とか「変数名の綴りの途中に予約語が含まれていてはいけない」といった制約のあるものが多い．本機にはそういう制約がないので非常に使い易い．

［注意］　長い変数名を使うのは，文法上は自由であるが，じつは計算が少し遅くなる．たとえば，1.2節の1）に示した所要時間3秒は1文字の変数名（**I**，**S**など）で計算した場合の値であるが，同じプログラムを10字の変数名に変えて計ってみると5秒かかる．

3.6 式の書き方

演算子　加減乗除を

+ − * /

で表す．ベキ乗は^印で表す．

(例)　X^Y　は　**X^Y**　で表す．

カッコ　丸カッコ(　)だけが使用可能．何重に使ってもよい．

(例)　$\dfrac{1}{x+\dfrac{1}{x+1}}$　は　**1/(X+1/(X+1))**　で表す．

計算順序　普通の数式と同様に

カッコの中を先に計算する

加減算よりも乗除算を先に計算する

さらにベキ乗があれば先に計算する

という規則で処理される．なお優先順位が同じものについては

左から順に計算される

のが普通である．

(例)　**A/B/C**　は　**(A/B)/C**　と同じ結果になる．

[**備考**]　上記以外の記号，たとえば

! % # $ ¥

などは，普通の数式における意味（階乗，パーセントなど）には使えない．特に注意を要するのは**!**および**%**で，これは後述のようにそれぞれ実数型および整数型を表す属性文字であるから，たとえば

5!　**5%**

は「5の階乗」「5パーセント」ではなく「実数型の5」「整数型の5」を表す．

整除（商が整数で表せる所まで割る）は**¥**印，**余り**は**MŌD**で表す．綴り**MŌD**の左は原則として1字あけること．

(例)　**C=A¥B**　　AをBで整除した商をCとする．

R=P MŌD 2　　Pを2で割った余りをRとする．

例題 1 ―― 和差積商(1)

二つの数値を読み込み，その和，差，積，商を計算して表示するプログラムを作れ.

[解答] 和，差，積，商を変数名**WA, SA, SEKI, SYŌU**で表すことにすれば，プログラムは次のようになる.

```
10 INPUT A,B
20 WA=A+B
30 SA=A-B
40 SEKI=A*B
50 SYOU=A/B
60 PRINT WA,SA,SEKI,SYOU
```

例題 2 ―― 円の面積

半径 r の値を読み込み，円の面積 $S=\pi r^2$ を計算して出力するプログラムを作れ.

[解答] 本機では π という文字が円周率を表すので，それを用いるとよい. π は GRAPH + A で入力する.

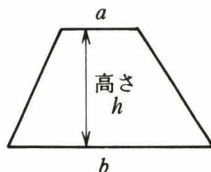
```
10 INPUT R
20 S=π*R^2
30 PRINT S
```

```
RUN
? 2
 12.566371
OK
```

演 習 問 題

2.1 (台形の面積) 台形の底辺の長さ a，b および高さ h を読み込み，面積

$$S=\frac{1}{2}(a+b)h$$

を計算して出力するプログラムを作れ.

3.7　組込み関数

一般に，BASIC では次の関数を使用できる（注1）.

種類	数学記号	BASIC における書き方	
平方根	$\sqrt{x}$	SQR(引数)	
指数関数	e^x	EXP(引数)	
自然対数	$\log_e x$, $\ln x$	LŌG(引数)	
正弦	$\sin x$	SIN(引数)	(注2)
余弦	$\cos x$	CŌS(引数)	(注2)
正接	$\tan x$	TAN(引数)	(注2)
逆正接	$\arctan x$	ATN(引数)	(注3)
絶対値	$\|x\|$	ABS(引数)	
整数化	$[x]$	INT(引数)	(注4)
符号		SGN(引数)	(注5)

注1）このほかに，

文字列処理のための関数

乱数発生のための関数

状態を調べるための関数

などがある.

注2）引数の単位はラジアンとして扱う．度をラジアンに直すには

$\pi/180 \fallingdotseq 0.0174533$

を掛ければよい.

注3）単位はラジアン．値の範囲は$-\pi/2 \sim \pi/2$になる.

ディスク BASIC でないと使用できない.

注4）xを越えない最大の整数.

(例)　INT(3.14)は3　　　INT(-3.14)は-4になる.

注5）$x<0$　ならば　SGN(X)の値は　-1　になる

$x=0$　ならば　〃　0　〃

$x>0$　ならば　〃　+1　〃

[備考]　関数の精度は約8桁と考えてよい．上記以外の数学的関数は組込まれていないので自作する必要がある.

(例)　立方根$\sqrt[3]{x}$は　X^(1/3)　として計算

arcsin x は　ATN(X/SQR(1-X*X))　として計算

以上のほかに，本機では特に下記の関数を使用できる．

	本機の BASIC による書き方
階乗（n!）	**FAC（引数）**注1)
n までの整数の和	**SUM（引数）**注2)
度をラジアンに変換	**RAD（引数）**
円周率 π の引数倍	**PAI（引数）**注3)
整数部	**FIX（引数）**注4)
小数部	**FRAC（引数）**注5)

（例）

```
10 INPUT N
20 H=SUM(N)
30 K=FAC(N)
40 PRINT N,H,K
RUN
? 10
 10        55        3628800
OK
```

注1）引数は0〜33の整数でなければいけない．

注2）$1+2+3+\cdots+n=\dfrac{n(n+1)}{2}$ の値が求められる．

三角マトリックスの圧縮格納等に便利である．

注3）たとえば**PAI（2）**は 2π

注4）正数の場合は**INT**と同じ結果になるが，負数の場合は異なる．

（例）**INT（−3.14）**　の値は　**−4**

FIX（−3.14）　の値は　**−3**

注5）符号と小数部を残して整数部を消す．

（例）**FRAC（3.14　）**　の値は　**0.14**

FRAC（−3.14）　の値は　**−0.14**

[注意]　これまでのシャープのパソコン（MZシリーズ）では，**LŌG（x）**は常用対数を表し，自然対数は**LN（x）**で表していたが，本機では他社のパソコンと同様に**LŌG**が自然対数を表すようになり，**LN**は廃止された．本機でうっかり**LN（x）**を使うと「**LN**という名前の配列の要素」と解釈されるから，値はたいてい0になる．既存のプログラムの利用に際しては十分に注意されたい．

例題 3 ―― 2次方程式の根(1)

2次方程式

$$ax^2+bx+c=0$$

の根を求めるプログラムを作れ.

[解答] 根の公式を用いる.

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

```
10 INPUT A,B,C
20 X1=(-B+SQR(B^2-4*A*C))/(2*A)
30 X2=(-B-SQR(B^2-4*A*C))/(2*A)
40 PRINT X1,X2
```

[注意] $b^2-4ac<0$ になるデーターを入れると

```
Illegal function call in 20
```

というエラー・メッセージが出て止まる.

例題 4 ―― 三角関数

$\theta=30^\circ$ に対する

$\sin\theta$ $\cos\theta$ $\tan\theta$

の値を計算して出力するプログラムを作れ.

[解答] 36ページの注2)を考慮して次のように計算する.

```
10 THETA=30
20 X=THETA*π/180
30 S=SIN(X)
40 C=COS(X)
50 T=TAN(X)
60 PRINT S,C,T
```

```
RUN
 .5         .8660254         .57735027
```

例題 5 ―― 常用対数

$\log_{10}x$ を計算するにはどうすればよいか.

[解答] LŌG(X)/LŌG(10)

例題 6 ――― 多項式の計算

x の値を読み込み

$$y = 2x^3+5x^2+3x+1$$

の値を計算して出力するプログラムを作れ.

[解答] 多項式

$$2x^3+5x^2+3x+1 \tag{1}$$

の値を計算するには，式を

$$((2x+5)x+3)x+1 \tag{2}$$

のように変形してプログラムに書くとよい.

```
10 INPUT "X=",X
20 Y=((2*X+5)*X+3)*X+1
30 PRINT "Y=",Y
```

もとの式(1)をそのまま使って

```
10 INPUT "X=",X
20 Y=2*X^3+5*X^2+3*X+1
30 PRINT "Y=",Y
```

と書いても誤りはないが，式(2)のように変形して計算する方が

計算時間が短い

計算誤差が少ない

などの利点がある.

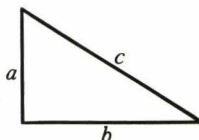
演 習 問 題

2.2 (立方根) 体積 V の立方体の1辺の長さ a を計算するプログラムを作れ.

[ヒント] $a = \sqrt[3]{V} = V^{(1/3)}$ を計算して出力すればよい（BASIC のベキ乗演算の指数部は整数でなくてもよく，そこに式を書いてもよい).

2.3 (ピタゴラスの定理, 三平方の定理) 直角三角形の直角をはさむ2辺の長さ a, b を読み込み，残る1辺の長さ

$$c = \sqrt{a^2+b^2}$$

を計算して出力するプログラムを作れ.

3.8 利用者が定義する関数

組込み関数（3.7節参照）以外の関数は，自分で作らなければならない．それを一つのプログラムの中の何箇所もで使うのであれば，サブルーチン（4.17節参照）の形で扱うのがよい．しかし，特に，関数の計算式が1行で書ける場合には，「利用者が定義する関数」という形で扱うと，

　　引数を設けることができる

　　算術式の中に書ける

という利点がある．

■ 関数を定義するには**DEF**文を用いる．これは次の形式で書く．

　　DEF　関数名（仮引数）＝その関数の計算式

■ 関数名は3文字以上で，

　　先頭2文字は**FN**に限る

　　第3字は英字に限る

　　FNの次に予約語と同じ綴りがあってはいけない

（正しい例）　**FNA　FNC**

（誤りの例）　**TAN　F　FNSINH**

■ 仮引数が不要ならば無くてもよい．その場合の形は

　　DEF　関数名＝その関数の定義式

となる．

■ 仮引数としては変数名を書く．本文の変数名と重複可．

■ 関数は，関数名（実引数）の形で代入文などの式の中に書くことができる．実引数としては，定数，変数のほか，式を書いてもよい．引数なしの関数は関数名だけで引用される．

◆ **DEF**文は，そこをプログラムの流れが通過するときに関数が「定義」される．単に「プログラムの一部に書いてある」だけでは有効にならないので注意を要する．

3.9　自分で簡単に作れる関数

下記の関数は組込み関数に入っていないが，**DEF**文によって容易に自作することができる．

$$\sinh x = \frac{e^x - e^{-x}}{2}$$

$$\cosh x = \frac{e^x + e^{-x}}{2}$$

$$\tanh x = \sinh x / \cosh x$$

$$\arcsin x = \arctan \frac{x}{\sqrt{1-x^2}}$$

$$\arccos x = \pi/2 - \arcsin x$$

$$\sqrt[3]{x} = x^{1/3}$$

[プログラム例]　上記の関数を BASIC で書いた例を示す．

```
10 INPUT "X=",X
20 DEF FNSH(X)=(EXP(X)-EXP(-X))/2
30 DEF FNCH(X)=(EXP(X)+EXP(-X))/2
40 DEF FNTH(X)=FNSH(X)/FNCH(X)
50 DEF FNASIN(X)=ATN(X/SQR(1-X*X))
60 DEF FNACOS(X)=π/2-FNASIN(X)
70 PRINT "X=";X
80 PRINT "sinh(x)=";FNSH(X)
90 PRINT "cosh(x)=";FNCH(X)
100 PRINT "tanh(x)=";FNTH(X)
110 PRINT "arcsin(x)=";FNASIN(X)
120 PRINT "arccos(x)=";FNACOS(X)
```

[出力例]

```
X= .5
sinh(x)= .5210953
cosh(x)= 1.127626
tanh(x)= .46211716
arcsin(x)= .52359878
arccos(x)= 1.0471976
```

3.10　PRINT 文

結果を表示するにはPRINT文を用いる．最も簡単な使用法は

PRINT　変数名　　または　　P．変数名

であるが，一般に式を書いてもよい．

PRINT　式　　または　　P．式

（例）
```
PRINT  X
PRINT  (A+B)*(C+D)/2
```

二つ以上列記したい場合はコンマまたはセミコロンで区切って書く．

（例）
```
PRINT  A,B,C+D
PRINT  A;B;C+D
```

セミコロンで区切った場合は，前の値にすぐ続けて（間をあけずに*）次の値が出力される．それに対しコンマで区切った場合は，一定の位置までスキップしてから次の値が出力される．**

（例）
```
10 A=12.3
20 B=5.67
30 C=-890
40 PRINT A,B,C
45 PRINT            これは間を1行あけるため
50 PRINT A;B;C
```

の実行結果は次のようになる．

```
RUN
 12.3          5.67          -890

 12.3  5.67 -890
OK
```

（上はコンマで区切った場合，下はセミコロンで区切った場合である）．

---

*　ただし数値データーの出力の場合，後に1字分の空白が入る．また先頭は符号であるが，＋印のかわりに空白が出力される．そのため正数の間には2字分の空白が入ることになる（上の例の12．3と5．67の間がそれである）．

**　位置は画面上では左端から10字間隔，プリンターでは8字間隔で設定されている．

改行と非改行　PRINT文と改行の関係は，「特に指定しない限り，1箇のPRINT文が出力の1行に対応する」というのが第1原則である．

(例)　AとBとCを1行目に，DとEを2行目に出力したければ

```
PRINT  A,B,C
PRINT  D,E
```

とすればよい．

同じ行に続きを出力したい場合は，PRINT文の最後にコンマまたはセミコロンを書いておく．左のページで述べた規則と同様に，コンマならば一定位置までスキップし，セミコロンならば間をあけずに(ただし前のページの脚注参照）続きが出力される．

(例)　
```
PRINT  A,
PRINT  B
```
は　PRINT　A,B　と同等．

逆に一つのPRINT文の途中で改行させることは，不可能ではないが*，プログラムの明快さを損ねるので，あまり使わない方がよい．

なお，以上のようにして定まる1行の文字数が画面の幅（1行に表示できる文字数）をオーバーする場合には，はみ出す直前の数値の区切りの所で自動的に改行される．

見出しの付け方　見出しなどのために文字列を出力したい場合は，出力したい文字列を2重引用符（"印）で囲んで，PRINT文の出力項目の所に書く．

```
10 Y=500
20 PRINT "金";Y;"円"
```

```
RUN
金 500 円
Ok
```

---

*　出力項目として改行コードCHR$(13)を書けばよい．

出力される数値の書式　符号や小数点を含めて10桁ぐらいでおさまる数値はそのままの（普通の）形で出力され，それでおさまらない数値は指数部付き形（±△．△△△△△E±△△）で出力される．

（例）　10, 100, …, 10^{15} とその逆数0.1, 0.01, …, 10^{-15} を出力すると次のようになる．

```
10                  .1
100                 .01
1000                .001
10000               .0001
100000              .00001
1000000             .000001
10000000            .0000001
1E+08               .00000001
1E+09               1E-09
1E+10               1E-10
1E+11               1E-11
1E+12               1E-12
1E+13               1E-13
1E+14               1E-14
1E+15               1E-15
```

また，$11^1, 11^2, \cdots, 11^{15}$ およびその逆数を出力すると次のようになる．

```
11                  9.0909091E-02
121                 8.2644628E-03
1331                7.513148E-04
14641               6.8301346E-05
161051              6.2092132E-06
1771561             5.6447393E-07
19487171            5.1315812E-08
2.1435888E+08       4.6650738E-09
2.3579477E+09       4.2409762E-10
2.5937425E+10       3.8554329E-11
2.8531167E+11       3.504939E-12
3.1384284E+12       3.1863082E-13
3.4522712E+13       2.8966438E-14
3.7974983E+14       2.6333125E-15
4.1772482E+15       2.3939205E-16
4.594973E+16        2.1762914E-17
```

［指数部の読み方］　±△．△△△△△E±△△は，±△．△△△△△×$10^{\pm\triangle\triangle}$ を表す．

（例）　**1E+06** は$10^6 = 1,000,000$

3.11 プリンターへの出力

プリンターに出力するには，PRINTのかわりにLPRINTと書けばよい．

例題 7 和差積商(2)

二つの数値を読み込み，その和，差，積，商を計算してプリンターに出力するプログラムを作れ．

[解答] 要するに例題1のプログラムのPRINTをLPRINTに書きかえればよいのであるが，単に結果の数値を四つ並べて印刷しただけでは，あとで意味がわからなくなってしまうから，なるべく

入力データも印刷する

それぞれの値に説明を付ける

ということを心掛けるとよい．以下に示すのは，そのようなプログラムの一例である．

```
10 INPUT "A=";A
20 INPUT "B=";B
30 LPRINT "A=";A
40 LPRINT "B=";B
50 LPRINT "A+B=";A+B
60 LPRINT "A-B=";A-B
70 LPRINT "A*B=";A*B
80 LPRINT "A/B=";A/B
```

[出力例]

```
A= 12
B= 3
A+B= 15
A-B= 9
A*B= 36
A/B= 4
```

演 習 問 題

2.4 **(会計)** 単価と数量をキーボードから入力し

(単価)×(数量)＝(金額)

の計算をしてプリンターに出力するプログラムを作れ．

3.12 REM

プログラムの中に注釈（説明，覚え書き）を書いておくと，あとで読むとき，わかり易くてよい．注釈はREM文の形で書く*．書き方は

REM　注釈

で，注釈としては任意の文字列（英数字，カナ文字，記号，空白など，何でも自由に使用できる）を書くことができる．

REM文はプログラムの中のどこに書いてもよい．ただし（後述の）マルチ・ステートメントに関しては注意が必要である（3.16節参照）．

（例）
```
10 REM **********************
15 REM *                    *
20 REM *      フクリ ケイサン      *
25 REM *                    *
30 REM *  N...ネンスウ          *
35 REM *  R...ネン リリツ (%)    *
40 REM *  A...ガンキン          *
45 REM *  G...ガンリ ゴウケイ     *
50 REM *                    *
55 REM **********************
60 REM データー ヨミコミ
65 INPUT "N=";N
70 INPUT "R=";R
75 INPUT "A=";A
80 REM ケイサン
85 G=A*(1+R/100)^N
90 REM ケッカ ヲ ヒョウジ
95 PRINT "G=";G
```

綴りREMのかわりに′印（アポストロフィー）を用いてもよい．

′　注釈

```
10 ' ジソク(km/h)ヲ ビョウソク(m/sec)ニ カンサン
20 '
30 INPUT "ジソク(km/h)=";V
40 U=V/3.6
50 PRINT "ビョウソク      ";U;"(m/sec)"
```

* REMはremarkの意味．

3.13 END と STOP と PAUSE

END, STŌP, PAUSEは，いずれも「止まれ」という命令であるが，機能がそれぞれ次のように異なる．

> **END**　これは「プログラムの実行はここで終る」という意味で，実行再開の必要のない場合（プログラムの終点）に用いる．

FORTRAN においては，**END**が「プログラムの最後の行」という特別な意味をもっているが，BASIC の**END**はそのような意味をもたない．したがって，**END**を何箇所に書いてもよく，一つも書かなくてもよい．これまでに例示したプログラムのように，**END**を書かなくても，行番号最大の文に来れば自然に実行終了となる．

> **STOP**　これは「プログラムの実行をここで一度中断する」という意味で，再開は
>
> 　　**CŌNT** ⏎
>
> で指示する．

STŌP文で停止すると「ピッ」と電子音が鳴り，

　　Break in STŌP文の行番号

というメッセージが表示される．

> **PAUSE**　これは「ここで一時停止し，指定時間経過後，自動的に実行を開始せよ」という文で，停止時間の長さは
>
> 　　PAUSE 停止時間の長さ
>
> （単位は0.1秒，値は最大255）の形で指定する．

PAUSE文は普通の BASIC にはない．ポケコン PC-1500 には**PAUSE**文があるが本機の**PAUSE**と機能が異なる．

3.14 整数演算および倍精度演算

（初心者はここをとばしてよい。）

本機の BASIC で扱うことのできるデーターの種類には，大別して

- 数値データー
- 文字データー

がある．この内，数値データーの扱い方に関しては

整数型	（演算が速く，記憶場所が少なくて済む）
実数型	（普通の扱い方）
倍精度実数型	（高精度の計算ができる）

の区別があり，文字データーに関しては，

文字列型　（英数字の文字列を可変長の形で扱う）

がある．

バイト　本機の内部でデーターを扱う際の最小単位は 8 ビット（2 進法の 8 桁）でこれを 1 バイト（byte）という．基本的な演算，記憶装置の出し入れ，周辺装置との間のデーターのやりとり，などはすべて(基本的には) 1 バイト単位で行なわれる．

1 バイト　[±｜｜｜｜｜｜｜]　8 ビット

整数型　整数型のデーターは 2 バイト（16ビット）の 2 進数で表現される．16ビットの内の 1 ビットは符号を表すのに用いられる．また負数は補数*の形で表現される．そのため，整数型で扱うことのできる値の範囲は

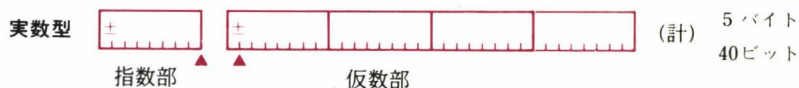
$$-32768 \sim +32767$$

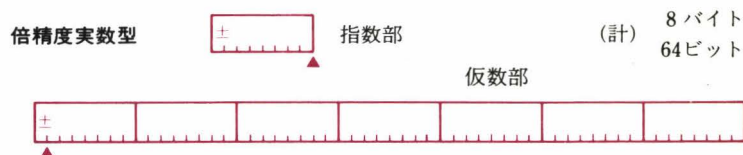
となる．（5 万円というような大金は整数型では扱えないので注意！）．

整数型　[±｜｜｜｜｜｜｜][｜｜｜｜｜｜｜｜]　2 バイト 16ビット

---

* いわば「ゲタをはかせて正数に直した形」のこと．本機の BASIC の整数型の場合，負数には 2^{16} =65536を加えて表している．こうすると演算処理が簡単になるのである．

実数型　本機の実数型は普通のパソコンよりも桁数が長い．実数型のデーターは，指数部（位どりを仮数部×$2^{(指数)}$の形で表す）1バイトと仮数部（有効数字を［±］０．＊＊＊＊＊＊＊の形で表す）4バイトの合計5バイトで表現され，処理される．やはり2進法である．

表現できる数値の範囲（絶対値）は$(1/2)\times2^{-128}\fallingdotseq10^{-39}$から，$(1-2^{-23})\times2^{127}\fallingdotseq10^{38}$まで，および0で，有効数字は2進法31桁であるから10進法換算約8桁である．

倍精度実数型　倍精度実数型のデーターは，指数部1バイト，仮数部7バイト，合計8バイトで扱われる．これも2進法である．

表現できる数値の範囲（絶対値）は約10^{-39}から約10^{38}まで，有効数字（仮数部の長さ）は2進法55桁であるから10進法換算約16桁である．

組込み関数の倍精度演算　本機の組込み関数（SIN，CŌS，LŌG，EXP，SQRなど）は，引数が倍精度型ならば関数も倍精度で計算される．

（例）

```
10 INPUT "X=";X#
20 LPRINT "x=";X#                x= .5
30 LPRINT "sin(x)=";SIN(X#)      sin(x)= .479425538604203
40 LPRINT "cos(x)=";COS(X#)      cos(x)= .8775825618903727
50 LPRINT "tan(x)=";TAN(X#)      tan(x)= .5463024898437905
60 LPRINT "exp(x)=";EXP(X#)      exp(x)= 1.648721270700128
70 LPRINT "log(x)=";LOG(X#)      log(x)=-.693147180559945З
80 LPRINT "SQR(x)=";SQR(X#)      SQR(x)= .7071067811865475
```

3.15 属性文字と型宣言

変数の型の種別を指定する方法は

属性文字を付ける

型宣言による

の二通りある．前者の場合は，普通の変数名の後に次のような属性文字（型の種類を表す文字）を付ける．

整数型は	%	(例)	A%
実数型は	!	(例)	A!
倍精度実数型は	#	(例)	A#

後者の場合は，次のような型宣言文により，変数名の先頭の文字（頭文字）と型の対応関係を指定する．

整数型は	DEFINT	先頭の文字
実数型は	DEFSNG	〃
倍精度実数型は	DEFDBL	〃

(例) 「Iで始まる変数名はすべて整数型」ということを宣言するには

```
DEFINT I
```

と書けばよい．なお，次のような書き方もできる．たとえば

```
DEFDBL D,S,W
```

これは「DまたはSまたはWで始まる変数名はすべて倍精度実数型」ということを意味する．また

```
DEFINT I-N
```

というような書き方もできる．これは「アルファベットのIからNまでの文字で始まる変数はすべて整数型」ということを意味する．

特に型の指定がなければ，変数はすべて実数型として扱われる．型宣言と属性文字による指定が一致しない場合は属性文字が優先される．

数値定数の型を指定するには，数値の後に左記と同じ属性文字を付けるのが最も確実である．

(例)　5%　　整数型の5

5!　　実数型の5

5#　　倍精度実数型の5

数値定数は指数部を付けた形で書くことができる．書き方は

符号　数字の列．数字の列　E　符号　数字の列

または

符号　数字の列．数字の列　D　符号　数字の列

でそれらの

EまたはDより左が仮数部（いわば有効数字）

EまたはDより右が指数部

で，値　(仮数)×10$^{(指数)}$

を表す．このような形で書かれた定数の型は次のようになる．

Eを付ければ実数型

Dを付ければ倍精度実数型

代　入　どの型についても，これまでに説明したのと同じ形で，代入をしたり，配列を用いたり，入出力を行なったりすることができる．正式の書き方は

LET　代入先の変数名＝式

であるが**LET**を省略して

代入先の変数名＝式

と書いてよい．代入先の変数名の型と式の型が一致していれば全く問題ないが，一致していなくても，

整数型　　実数型　　倍精度実数型

の間では自動的に変換が行なわれる（まず，式の値を計算し，代入先の変数名の型に合わせて変換し，その上で代入を行なう）．整数型に変換する際，小数部は四捨五入される．

3.16 文法細則

以下で説明する事項は，知らなければ知らなくても済むことであるが，覚えておけば役に立つ．

変数の初期値は 0　　プログラムの実行開始時点において，すべての変数の値は 0 になっている．

(例)　何も代入しないで　**PRINT　A**　を実行すれば0が出力される．

余分な空白は無視　　本書では，プログラムを読を易くするため，文の中に空白をたくさん入れているが，BASIC では（**REM**文および2重引用符で囲まれた部分を除き）空白は文法上，特別な意味をもたない（無視される）ということになっている．

ただし，予約語（たとえば**MŌD**，**THEN**，**AND**など）のすぐ左に変数名を書くときは間に1字以上の空白を入れる必要がある．

(例)　**PRA　MŌD　EL**　は「**PRA**÷**EL**の余り」

PRAMŌDEL　は「**PRAMŌDEL**という名前の変数」

省略記法　　記号？は命令の略記法に用いられ，**PRINT**を表す．

(例)　?　**A，B，C**

?印は直接実行形式の場合によく用いられる．たとえば，変数**A**の内容（現在記憶されている値）を調べたいときに

?　**A**　⏎

とやって表示させたり，$\pi/3$ の値を知りたいときに，電卓がわりに

?　π / 3

というぐあいに用いたりする．

マルチ・ステートメント　一つの行に二つ以上の文を書くことができる．これをマルチ・ステートメントという．二つ以上の文を書く場合は，文の間をコロンで区切る．

(例)
```
10  INPUT  A,B  :  PRINT  A+B
```
行番号　一つの文　区切り記号　もう一つの文

[解説] 本書では，これまで常に，一つの行に一つの文を書いてきた．それが伝統的なBASICの標準的な書き方である．1行1文ならば，すべての文に行番号が付いており，それによって編集したり，行先を指示(4章で説明する)したりすることができる．初心者はなるべくそういう形で書く方がよいと思う．

しかし，少し慣れてきて，ある程度長いプログラムを作るようになると，1行にいくつもの文を書きたくなる．それは次のような理由による．

1）ディスプレイの1画面になるべくたくさんの文を表示できる方がデバックをし易い．

2）処理の内容によっては，横に並べて書く方が自然な場合がある．たとえば，複素数の計算をするとき，実数部の処理と虚数部の処理を並行して書くと見易い．

3）他のプログラミング言語ならば一つの文で書けるのにBASICだと複数箇の文でないと書けないことがある．そういうとき，一つの機能的まとまりを示すために1行で書くと見易くなる．

4）1行1文にするよりも，マルチ・ステートメントにする方が，記憶場所をいくらか節約できる．

5）文の挿入の一手段として使える．

[注意] REM文と後述のIF文に関しては「行の終りまでを一つの文とみなす」という扱いになっているので，REM文やIF文の右にマルチ・ステートメントの形で他の文を書くことはできない．

(例)
```
10  REM  "example"  :  INPUT  A
```
の「INPUT　A」の注釈の一部とみなされる．

小数に関する注意　　整数の加減算および乗算は２進法で計算しても10進法で計算しても全く同一の結果が得られるが，小数に関しては，２進法で計算した結果と10進法で計算した結果が（有限桁数で計算すると）ごくわずかではあるが違ってくるので注意を要する．

原因は10進法の0.1が２進法だと

0.000110011001100110011001100…

という無限小数になり，有限桁数では正確に表現できないからである．そのため，たとえば0.1を10箇加えても，ちょうど１にはならない．

（例）
```
10 A=.1
20 X=A+A+A+A+A+A+A+A+A+A
30 E=X-1
40 PRINT E
```

```
RUN
 4.6566129E-10
```

4 制御文——枝分かれとくりかえし

4.1 概　　説

BASICの文は原則として行番号の順に実行されるが,

　　ある行までスキップしたい（または，ある行にもどりたい）

　　条件に従って分岐し，それぞれ別の処理をしたい

　　プログラムの一部分をくりかえし実行したい

という場合がある．そのために，BASICには次のような文がある．まず，指定した行に進む文としては

　　GŌTŌ文 ……………………（行きっぱなし）

　　GŌSUB文 …………………行ってもどってくる

分岐に関しては

　　IF文 ………………………条件式によって分岐

　　ŌN文 ………………………番号によって分岐

くりかえしに関しては

　　FŌR～NEXT文 …………一定回数の反復

　　REPEAT～UNTIL文 …条件式が成立するまで反復

　　WHILE～WEND文 ………条件式が成立している間反復

がある．本機では，GŌTŌ文，GŌSUB文，IF文，ŌN文などの行先の指定にラベル (label) を用いることができる（もちろん普通のBASICと同様に行番号で指定することもできる)．ラベルは英字で始まる英数字の綴りを2重引用符で囲んで表す．

(例)　"SAPPŌRŌ"

4.2 GOTO文

無条件に特定の行までスキップしたり，もどったりするにはGŌTŌ文（英語の“Go to～.” の意味であるから [goutu:] と読む．BASICではGŌとTŌの間に空白を入れないのが普通）を用いる．書き方は

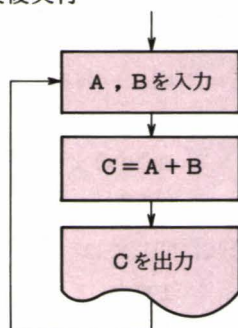
GŌTŌ　行先の行番号　または　ラベル

（例）　GŌTŌ　100　　「行番号100の文に行け」という意味

GŌTŌ　"SAPPŌRŌ"

GOTO文の応用(1)　―― 同じプログラムの反復実行 ――

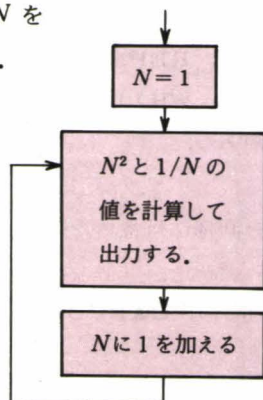
プログラムの最後の所から最初にもどるようにGŌTŌ文を書いておけば，停止ボタンを押すまで何回でも反復実行できる．

```
10 INPUT "A=";A
20 INPUT "B=";B
30 C=A+B
40 PRINT "A+B=";C
50 GOTO 10
```

GOTO文の応用(2)　―― 値を少しずつ変えて反復 ――

（例）　Nの値を1，2，3，…と変えてN^2と$1/N$を計算して出力するには，次のようにすればよい．

```
10 N=1
20 S=N^2
30 R=1/N
40 PRINT N,S,R
50 N=N+1
60 GOTO 20
```

（別解）

```
10 N=1
20 LABEL "KURIKAESI"
30 S=N^2
40 R=1/N
50 PRINT N,S,R
60 N=N+1
70 GOTO "KURIKAESI"
```

プログラムを途中から始める方法　エラーなどでプログラムが停止したとき，中間結果を保持したまま，途中の文から実行を再開させることもできる．そこにはGŌTŌ文を直接実行形式で用いて

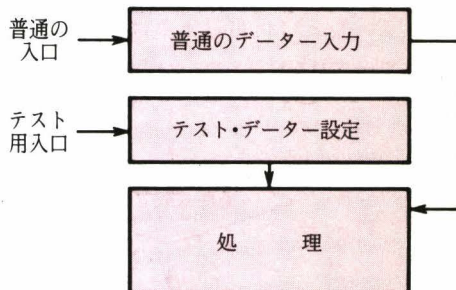
GŌTŌ　実行開始行番号　または　GŌTŌ　ラベル

とすればよい．たとえば，

GŌTŌ　300 ⏎

と指令すれば，行番号300の所から実行が再開される．

GOTO 文の応用(3)　―― 入口を二つ作る ――

(例)　プログラムの普通の入口のほかに「テスト用の入口」を設け，いちいちテスト・データーを入力しなくても標準的なテスト・データーが設定されるようにしておくと便利である．

```
10 REM --- normal entry ---
20 INPUT A,B,C
30 GOTO 60
40 REM --- special entry for program test ---
50 A=2 : B=10 : C=12
60 REM --- computation ---
70 X=(-B+SQR(B^2-4*A*C))/(2*A)
80 PRINT X
```

RUN　50 ⏎ で開始すると，方程式

$$2x^2+10x+12=0$$

の根が計算され，出力される．

ラベルの付け方　ラベルはLABEL文で書く．書き方は

LABEL　"ラベル"

右に文を書くときは文との間に：印を入れる．マルチ・ステートメントの途中に書くこともできる．

4.3 IF文による分岐

ある条件のときだけ，指定した所に進むようにするには，次のように書く．

IF　条件式　THEN　行先

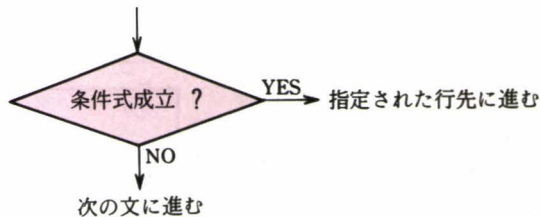
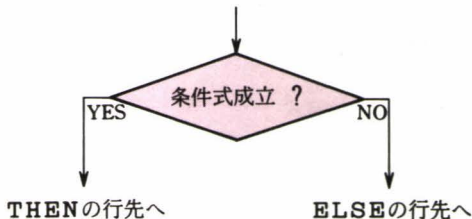
これを次のように書くこともできる．

IF　条件式　GŌTŌ　行先

条件が成立しない場合の行先を指定することもできる．書き方は

IF　条件式　THEN　行先　ELSE　行先

である．行先は行番号またはラベルで指定する．

IF〜THEN，IF〜GŌTŌ　概念図

IF〜THEN〜ELSE　概念図

[注意]　THENやELSEやGŌTŌの左には原則として1字以上の空白を入れなければいけない．

(例)　IF A=B GŌTŌ "SAPPŌRŌ"

　　IF C<D THEN 300 ELSE 500

4.4　条件式の書き方

■　条件式としては次のようなものが書ける.

1）二つの式を等号または不等号で結んだもの.

(例)　**A=B　　M+1>N　　ABS(E)<0.01**

ただし，等号，不等号は次のように書く.

	普通の数学記号	BASICの書き方
等しい	=	=
等しくない	≠	<>　または　><
より小さい	<	<
等しいかまたは小さい	≦	<=　または　=<
等しいかまたは大きい	≧	>=　または　=>
より大きい	>	>

2）等式または不等式（すなわち上記1)の形）に（必要ならば）**NŌT**を付け，それらを**AND**や**ŌR**で結んだもの．ただし，

AND	は「かつ」	両方とも成立すること
ŌR	は「または」	少なくとも一方が成立すること
NŌT	は「でない」	否定

を表す.

(例)　**A=0　ŌR　B=0**

3）上記2)の形をカッコで囲み，(必要ならば)**NŌT**を付け，それらを**AND**や**ŌR**で結んだもの．なお，カッコを省略した場合には

NŌT　AND　ŌR

の順に結合される.

(例)　**NŌT　A　ŌR　B　AND　C**　は次式と同等

(NŌT　A)　ŌR　(B　AND　C)

[注意]　**AND，ŌR，NŌT**の左は必ず1字以上の空白を入れなければいけない．ただし隣接する文字がカッコの場合は空白を入れないでよい.

例題 1　**2次方程式の根（2）**

2次方程式

$$ax^2 + bx + c = 0$$

の根（実根の場合もあり複素根の場合もある）を計算して出力するプログラムを作れ.

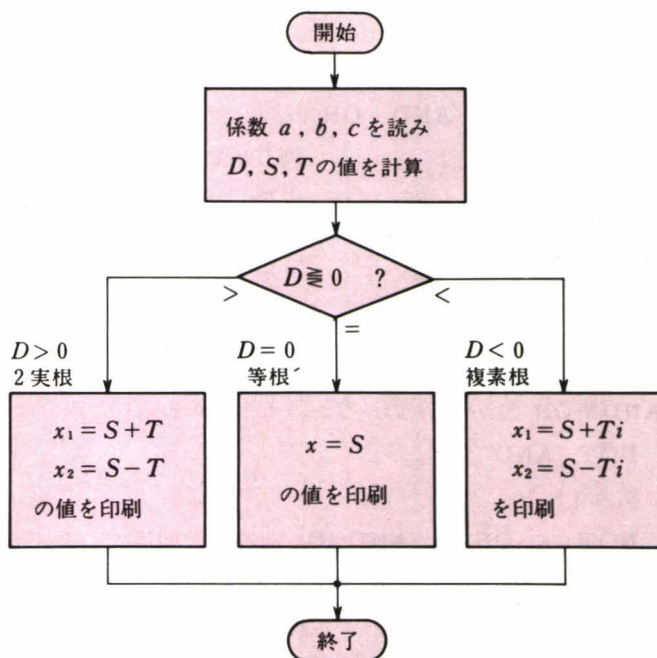
［解答］　2次方程式の根の公式を用いる. まず判別式

$$D = b^2 - 4ac$$

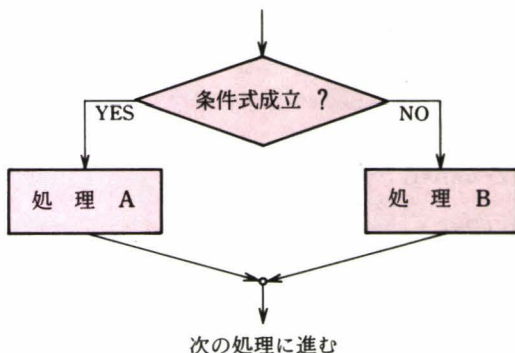
の値を計算し，ついでに，あとの便宜を考えて

$$S = -b/(2a) \qquad T = \sqrt{|D|}\,/(2a)$$

の値を計算しておいて，次のように処理する.

［流れ図］

```
10 REM ┌─────────────────────┐
20 REM │                     │
30 REM │  2ｼﾞﾎｳﾃｲｼｷ          │
40 REM │   A*X^2+B*X+C=0     │
50 REM │  ﾉ ｺﾝ ｦ ﾓﾄﾒﾙ。       │
60 REM │                     │
70 REM └─────────────────────┘
80 REM
90 PRINT "ｹｲｽｳ ｦ ｲﾚﾃ ｸﾀﾞｻｲ。"
100 INPUT "A=",A
110 INPUT "B=",B
120 INPUT "C=",C
130 PRINT "ｶｲ";
140 REM --- ﾊﾝﾍﾞﾂ ｼｷ ---
150 D=B^2-4*A*C
160 REM --- ﾀﾞｲ 1 ｺｳ ---
170 S=-B/(2*A)
180 REM --- ﾀﾞｲ 2 ｺｳ ---
190 T=SQR(ABS(D))/(2*A)
200 REM --- ﾊﾞｱｲ ﾜｹ ---
210 IF D>0 GOTO 290
220 IF D=0 GOTO 340
230 REM --- D<0 ﾉ ﾊﾞｱｲ(ｷｮｺﾝ) ---
240 PRINT "(ｷｮｺﾝ)"
250 PRINT "X1=";S;"+";T;"i"
260 PRINT "X2=";S;"-";T;"i"
270 END
280 REM --- D>0 ﾉﾊﾞｱｲ(ｼﾞｯｺﾝ) ---
290 PRINT "(ｼﾞｯｺ ﾝ)"
300 PRINT "X1=";S-T
310 PRINT "X2=";S+T
320 END
330 REM --- D=0 ﾉﾊﾞｱｲ(ﾄｳｺﾝ) ---
340 PRINT "(ﾄｳｺﾝ)"
350 PRINT "X=";S
360 END
```

4.5 IF文による場合分け処理

プログラムの流れは，枝分かれして，それぞれの処理を行なって，また合流する，という形がよくある．

そのような場合，処理が比較的簡単（短いプログラムで書ける）ならばそれをIF文の中に書くことができる．書き方は，

IF　条件式　THEN　処理A　ELSE　処理B

またはその後半（ELSE以後）を除いた形

IF　条件式　THEN　処理A

である．ここで処理Aと書いたのは「条件式が成立したときに実行すべきプログラム」，処理Bと書いたのは「条件式が成立しなかったときに実行すべきプログラム」のことで，実際には

一つの文

または，マルチ・ステートメント，すなわち

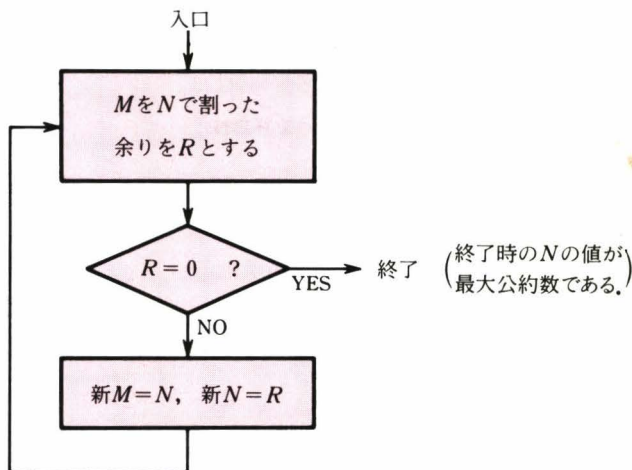
いくつかの文をコロン（：印）で区切って1行に書いたもの

の形で記述する．

(例)
```
IF X<0 THEN Y=0 ELSE Y=X
IF N=1 THEN I=0 : J=0 : K=0
IF A<B THEN PRINT A ELSE PRINT B
```

例題 2　**最大公約数**

正の数 M, N の最大公約数を計算するプログラムを作れ．ただし $M \geqq N$ とする．

[解答]　ユークリッドの互除法を用いるとよい．その手順を流れ図の形で書くと次のようになる．

```
10 INPUT "M,N=";M,N
20 R=M MOD N
30 IF R=0 THEN PRINT "GCD=";N : END
40 M=N : N=R
50 GOTO 20
```

演　習　問　題

4.1　MとNの最小公倍数を求めるプログラムを作れ．

[ヒント]　次の計算式を用いればよい．

$$(\text{最小公倍数}) = M \cdot N / (\text{最大公約数})$$

ただし上のプログラムでは M と N の値を計算の途中で書きかえてしまっているから，その計算に入る前に M と N の値（または積 MN の値）をどこかに記憶させておく必要がある．

例題 3 ―― 10行ごとに停止

$n = 1, 2, 3, \cdots$について

$$\left(1+\frac{1}{n}\right)^n$$

の値を計算してディスプレイに表示するプログラムを作れ．表示の際，10行ごとに停止させ，ゆっくり見られるようにせよ．

[解答] $n = 10, 20, 30, \cdots$，すなわち10の倍数のときに停止させればよい．nが10の倍数であるかどうかを調べるには

```
IF (N MOD 10)=0 THEN …
```

（N MOD 10：nを10で割った余り）

で判定すればよい．

```
10 N=1
20 A=(1+1/N)^N
30 IF (N MOD 10)=0 THEN STOP
40 PRINT N,A
50 N=N+1
60 GOTO 20
```

停止後の実行再開は

```
CONT ↵
```

（または，SHIFT + F5）によって行なう．

[別解] 次のようにすれば，いちいち再開の操作をしないで済む．

```
10 N=1
20 A=(1+1/N)^N
30 PRINT N,A
40 IF (N MOD 10)=0 THEN PAUSE 50
50 N=N+1
60 GOTO 20
```

IF 文の演習問題

4.2　次の内，IFの書き方として正しいものはどれか．

```
IF  A=B+C  GŌTŌ  30
IF  (A=B)  GŌTŌ  30
IF  NŌT  A=B  GŌTŌ  30
```

4.3　**(妥当性のチェック)**　年齢を入力したとき，その値が「あり得ない値」(たとえば負数とな，200以上など) でないかどうかを調べるプログラムを作れ．

4.4　**(例外処理)**　例題1のプログラムを次のように改造せよ．

1) $a = 0, b \neq 0$　でも計算できるようにする．

2) $a = b = 0, c \neq 0$　でも計算できるようにする．

3) $a = b = c = 0$ ならば，エラー・メッセージを出す．

4.5　**(大口割り引き)**　ある店のカセット・テープの値段は次のようになっている．

バラ（1本単位）だと，330円（1本につき）

1箱（10本入り）は，3000円（1箱につき）

n 本の値段を計算するプログラムを作れ．

4.6　**(機能選択)**　三角形の面積を計算する公式はいろいろあるが，まず公式番号 n を読み，

$n = 1$ ならば，2辺の長さ a, b とそのなす角 θ より

$S = (ab \sin \theta)/2$

$n = 2$ ならば，底辺の長さ a と高さ h より

$S = ah/2$

$n = 3$ ならば，3辺の長さ a, b, c より，ヘロンの公式

$S = \sqrt{s(s-a)(s-b)(s-c)} \qquad s = (a+b+c)/2$

で面積 S を計算するプログラムを作れ．

4.6 FOR～NEXT

一定回数のくりかえしを行なうにはFŌR文を用いる．これは次の形式で書く．

　　FŌR　制御変数名＝始値　TŌ　終値　STEP　増分

これは「制御変数の値を始値から終値まで増分ずつ変えて，以下の部分をくりかえし実行せよ」という意味になる．増分1の場合は省略して

　　FŌR　制御変数名＝始値　TŌ　終値

と書いてもよい．くりかえす部分の最後には

　　NEXT　制御変数名

を書く．これは「制御変数の値を次の値（すなわち，現在の値プラス増分）に変えて，くりかえし部分の最初にもどれ．ただし，制御変数の新しい値が終値を越えたならば，くりかえしを終了してNEXTの次の文に進め」ということを表す．なお

　　始値，終値，増分は必ずしも整数でなくてもよい*

　　始値，終値，増分として変数や式を書いてもよい

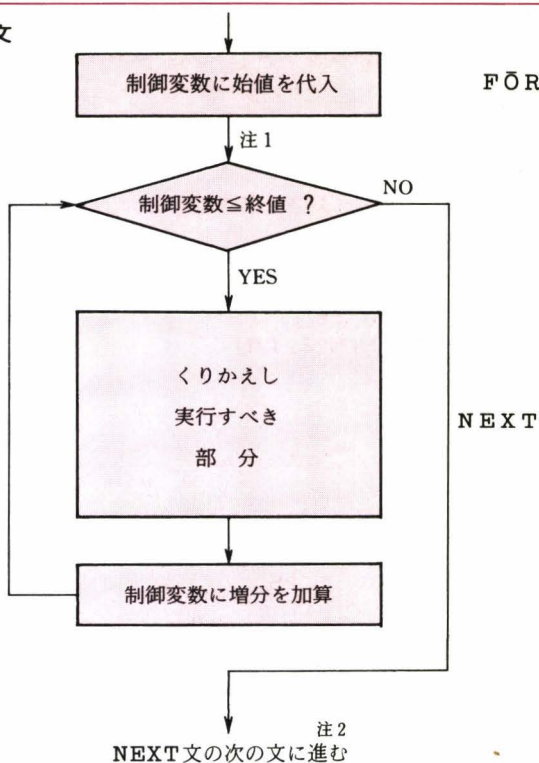
　　増分は負でもよい

(例)　Nの値を1から5まで変えて　$N(N+1)/2$　の値を計算し出力するには次のように書けばよい．

```
10 FOR N=1 TO 5
20     PRINT N,SUM(N)
30 NEXT N
```

```
RUN
 1           1
 2           3
 3           6
 4           10
 5           15
OK
```

---

*　ただし整数でないと，丸め誤差のために，くりかえし回数が，意図したとおりにならないことがあるので，なるべく整数を用いる方が安全である．

注1） 増分が負の場合には，終了判定の不等式の向きが逆になる．

注2） このとき制御変数の値は，「最後に実行したときの値プラス増分」になっている．

(例)

```
10 INPUT "N1=",N1
20 INPUT "N2=",N2
30 FOR N=N1 TO N2
40     PRINT N
50 NEXT N
60 PRINT "ｼｭｳﾘｮｳｺﾞ";N
```

```
RUN
N1=5
N2=7
 5
 6
 7
ｼｭｳﾘｮｳｺﾞ 8
OK
```

```
RUN
N1=2
N2=1
ｼｭｳﾘｮｳｺﾞ 2
OK
```

例題 4 数表を作る

1から10までの整数の2乗と逆数の数表を作成するプログラムを作れ.

[解答] Nの値を1から10まで変えて

$N \quad N^2 \quad 1/N$

の値を計算し出力すればよい.

```
10 PRINT "N","N^2","1/N"
20 FOR N=1 TO 10
30     PRINT N,N^2,1/N
40 NEXT N
```

[出力例]

```
1       1       1
2       4       .5
3       9       .33333333
4       16      .25
5       25      .2
6       36      .16666667
7       49      .14285714
8       64      .125
9       81      .11111111
10      100     .1
```

[備考] このプログラムだと,上の出力例のように,整数が左づめになり,右に大きな空白ができる.右づめにして適切な桁数で出力するには**PRINT USING**文(6.6節参照)を用いる必要がある.

FOR 文の演習問題

4.7 (数表を作る) 次の数表を出力するプログラムを作れ.

i) $\theta = 0^\circ$ から90° まで10° おきの

$\sin\theta \qquad \cos\theta \qquad \tan\theta$

ii) $x = 0.1$ から2.0まで0.1おきの常用対数 $\log_{10} x$.

iii) $n = 1$ から10までの $n!$.

4.8 (料金早見表) ガソリン・スタンド用の料金早見表を作りたい. 単価を4種類(製品の種類などによって異なる. たとえば, リッターあたり138円, 145円, 155円, 170円)読み込み, 1リッターから0.1リッターおきに30リッターまでの料金表を出力するプログラムを作れ.

4.9 (複利計算) 元金1万円を複利で運用したら1年後, 2年後, …, 20年後いくらになるか (元利合計) の表を作りたい. 3段階の年利率 (たとえば, 3.5%, 5%, 6%) を読み込み, 上記の表を出力するプログラムを作れ.

4.10 (積立預金) 毎年1万円を積立預金したら, 1年後, 2年後, …, 20年後の元利合計がいくらになるかについて, 前問と同様の表を出力するプログラムを作れ.

4.11 (年金) 退職金1000万円を原資とし, 毎年100万円ずつ引き出して使用したとして, 10年後までの毎年の元利合計を計算するプログラムを作れ. ただし預入れ時点には引き出さず, 1年後から引き出すものとする. また, 利率は問題4.9と同様に3段階の値を指定できるようにする.

4.12 (極限値) 数値計算によって数列の極限値の厳密な値を求めることはできないが, 極限値に近い値をいろいろ計算してみることによって極限値を推測することはできる. たとえば $\alpha = \lim_{x \to 0} x^x$ の値を知るには, $x_n = 10^{-n}$ に対する $\alpha_n = x_n^{x_n}$ の値を $n = 1, 2, 3, \cdots$ に対して計算してみればよいであろう. $n = 1$ から10までに対し, 上記 α_n の値を計算して出力するプログラムを作れ. また $\alpha = \lim_{x \to 0} \dfrac{\sin x}{x}$ について同様なプログラムを作れ.

4.7 多重のループ

FŌR〜NEXTによるループ（loop，くりかえし）は，何重にも入れ子の形で行なうことができる．

例題 5　組合せ

Ｉは１から５まで，Ｊは１からＩまでの全部の組合せについて，分数Ｊ／Ｉを小数になおして表示せよ．

[解答]

```
10 FOR I=1 TO 5
20    FOR J=1 TO I
30       PRINT J;"/";I;"=";J/I
40    NEXT J
50    PRINT
60 NEXT I
70 END
```

```
RUN
 1 / 1 = 1

 1 / 2 = .5
 2 / 2 = 1

 1 / 3 = .33333333
 2 / 3 = .66666667
 3 / 3 = 1

 1 / 4 = .25
 2 / 4 = .5
 3 / 4 = .75
 4 / 4 = 1

 1 / 5 = .2
 2 / 5 = .4
 3 / 5 = .6
 4 / 5 = .8
 5 / 5 = 1

OK
```

[解説]　行10のFŌR文でＩの値を１から５まで変えている．これに対応するNEXT文は行60にあるから，行20〜50が，各Ｉの値について実行される．行20のFŌR文ではＪの値を１からＩまで変えている．これに対応するNEXT文は行40にあるから，行30の文はＩ，Ｊの全部の組合せについて実行される．行30では，まずＪの値を出力し，続けて／印，次にＩの値，次に＝印，最後にＪ／Ｉの値を出力する．

4.8 配　　列

一組のデーターに番号を付けて規則的に並べたものを配列 (array) という．これは要するに表（ひょう）のことであるが，コンピューターの用語では配列というのが普通である．

データーを一列に並べたものを 1 次元配列という．

データーを四角に並べたものを 2 次元配列という．

データーを立体的に並べたものを 3 次元配列という．

……………………　（以下同様に何次元でも可）

配列名　配列には名前を付けて扱う．この名前を配列名という．配列名の付け方（使用文字や長さについての制限など）は変数名の付け方と同じで，先頭は英字，そのあとは英字または数字であればよい．

4.9　添字付き変数

配列に属する個々のデーターのことを**配列要素**という．配列要素は，配列名に添字（いいかえれば番号）を付けた形（これを**添字付き変数**という）で参照することができる．その書き方は

配列名(添字)　　　　………1次元配列の場合

配列名(第1添字，第2添字)　………2次元配列の場合

(以下同様)である．2次元配列の場合，第1添字は行番号，第2添字は列番号を表す．添字に負数は使用できないが，0は使用してよい．

(例)　配列A

	1列目	2列目	3列目
1行目	1.8	7.3	8.6
2行目	5.2	2.6	4.7
3行目	3.5	0.1	9.8

配列X

1番目	2.9
2番目	0.8
3番目	6.6
4番目	1.5

の場合，A(2,3)の値は4.7，A(3,2)の値は0.1，X(1)の値は2.9，A(1,3)+X(2)の値は8.6+0.8=9.4になる．

添字式　　添字としては整定数のほか変数名や式を書いてもよい．

(例)　A(I,J)　　X(2*N−1)

添字として用いる式の形についての制限はなく，どんな式でも書ける．

(例)　C(I/J+SIN(X(N))+5.8)

添字に端数（小数点以下）がある場合は四捨五入になる．

引　用　　添字付き変数は普通の変数と全く同様に使用することができる．ただしFŌR文の制御変数としては使用できない．

(例)　式の中に書く　　A(I)+B

　　　代入先に指定　　A(6)=X*Y+Z

　　　入　力　　　　　INPUT P(3)

　　　出　力　　　　　PRINT Q(7)

4.10 配列の宣言

DIM 文　配列を使用する場合は，あらかじめ，

DIM 配列名（寸法）　………1次元配列の場合

DIM 配列名（寸法，寸法）………2次元配列の場合

……以下同様

などの形で，配列の宣言を行なっておく必要がある．ただし寸法が10以下の場合は宣言せずに配列を用いることができる．

上で寸法と書いたのは添字の上限のことで，コンピューターはその値に合わせて配列の記憶場所を割り当てる．寸法は，あらかじめわかっていればその値を定数として書くのであるが，

（例）　DIM A(3,3),X(4),C(137)

入力データーや計算の状況に合わせて寸法を決めたい場合には，ここに変数や式を書くこともできる．

（例）　DIM A(M,N),X(M+1),C(M+N)

［解説］　BASICのDIM文は実行文（すなわち，実行時に機能する文）であって，配列の記憶場所の割り当ては，プログラムの流れがそこに到達した時点において行なわれる．したがって，一つのプログラム中に同じ配列の宣言が2箇所以上あってもよい（重複して実行されることがなければ）．

4.11 配列の入出力

入　力　パソコンの BASIC には一つの命令で配列の全要素の入出力を行なえるような機能はないので，1要素ずつの入出力のくりかえしの形でプログラムを書く必要がある*.

（例）　小さな配列**の入力は次のようにするとよい.

```
10 DIM A(3,2)
20 FOR I=1 TO 3
30    PRINT I;"ｷﾞｮｳﾒ ｦ ｲﾚﾃｸﾀﾞｻｲ"
40    INPUT A(I,1),A(I,2)
50 NEXT I
```

この場合，入力データーは1行分ずつ（1行の中ではコンマで区切って）

a_{11},　a_{12}

a_{21},　a_{22}

a_{31},　a_{32}

の順に入れる．一方，大きな配列の入力は次のようにするとよい.

```
10 DIM A(30,50)
20 FOR I=1 TO 30
30    FOR J=1 TO 50
40       PRINT "A(";I;",";J;")=";
50       INPUT A(I,J)
60    NEXT J
70 NEXT I
```

これだと，入力すべきデーターの行番号，列番号が表示されるので，該当するデーターを入れていけばよい.

```
RUN
A( 1 , 1 )=? 947.2511
A( 1 , 2 )=? 262.5256
A( 1 , 3 )=? 233.8641
A( 1 , 4 )=? 256.3491
A( 1 , 5 )=? 941.3131
A( 1 , 6 )=?
```

---

*　大型コンピューターの BASIC にある **MAT READ** 文や **MAT PRINT** 文は，パソコンの BASIC では使用できない.

**　配列の1行分を画面1行に表示できる場合

出　力　出力も同様で，以下の要領でプログラムを書けばよい．

(例)　配列**A**がM行N列で，$N \leqq 5$の場合

```
100 FOR I=1 TO M
110     FOR J=1 TO N
120        PRINT A(I,J),
130     NEXT J
140     PRINT
150 NEXT I
```

(例)　配列**A**がM行N列で，$N > 5$の場合，4列分ずつまとめてプリンターに出力して，あとで大きな台紙にはり合わせるとよい．（**TAB**については124ページ参照）

```
100 FOR KARA=1 TO N STEP 4
110     MADE=KARA+3
120     IF MADE>N THEN MADE=N
130     FOR J=KARA TO MADE
140        LPRINT "J=";J;TAB(16*(J-KARA+1));
150     NEXT J
160     LPRINT
170     FOR I=1 TO M
180        FOR J=KARA TO MADE
190           LPRINT A(I,J);TAB(16*(J-KARA+1));
200        NEXT J
210        LPRINT
220     NEXT I
230     LPRINT
240 NEXT KARA
250 END
```

[出力例]

```
J= 1            J= 2            J= 3            J= 4
 1               .5              .33333333       .25
 2               1               .66666667       .5
 3               1.5             1               .75
 4               2               1.3333333       1
 5               2.5             1.6666667       1.25
 6               3               2               1.5
 7               3.5             2.3333333       1.75

J= 5            J= 6            J= 7
 .2              .16666667       .14285714
 .4              .33333333       .28571429
 .6              .5              .42857143
 .8              .66666667       .57142857
 1               .83333333       .71428571
 1.2             1               .85714286
 1.4             1.1666667       1
```

例題 6　九九の表

九九の表を作成するプログラムを作れ．

[解答]　右図のような配列を用意し，その i 行 j 列目の所に積 ij の値を書き込んで印刷する．ただし，普通の方法で出力すると上下がきれいにそろわない．きれいな形で出力するには下記のプログラム例のように **PRINT USING** 文を用いるとよい．(122ページ参照)

```
100 DIM A(9,9)
110 FOR I=1 TO 9
120   FOR J=1 TO 9
130     A(I,J)=I*J
140   NEXT J
150 NEXT I
160 FOR I=1 TO 9
170   FOR J=1 TO 9
180     PRINT USING" ###";A(I,J);
190   NEXT J
200   PRINT
210 NEXT I
```

（110～150）i を 1 から 9 まで変え，その中で j を 1 から 9 まで変えて，積 ij を計算する．

これで配列 **A** の中に九九の表ができた．

（200）これは「改行せよ」という意味になる．

```
run
  1  2  3  4  5  6  7  8  9
  2  4  6  8 10 12 14 16 18
  3  6  9 12 15 18 21 24 27
  4  8 12 16 20 24 28 32 36
  5 10 15 20 25 30 35 40 45
  6 12 18 24 30 36 42 48 54
  7 14 21 28 35 42 49 56 63
  8 16 24 32 40 48 56 64 72
  9 18 27 36 45 54 63 72 81
```

4.12 合　　計

n 箇のデーター a_1, a_2, …, a_n を読み込み,

$$S = a_1 + a_2 + \cdots + a_n$$

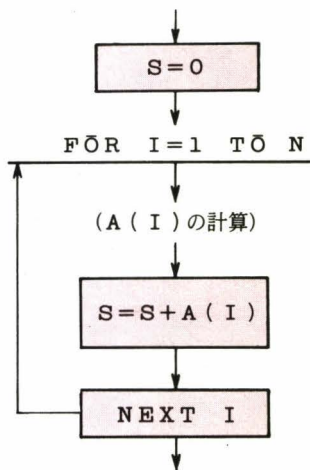
を計算して印刷するには次のようにする.

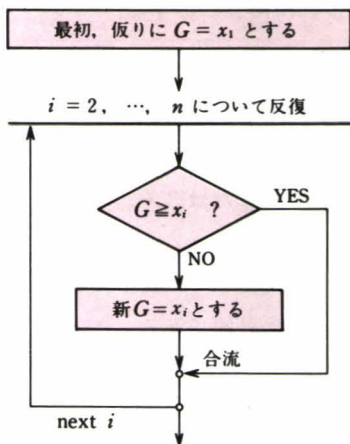
1）部分和を表す変数を設け,
　その値（出発値）を 0 にしておく.
2）第 1 項, 第 2 項, …の順に
　　項の値を計算（または読み込む）
　　項の値を部分和に加える
　　最後の項まで反復
3）部分和の最終値が合計値である.

[プログラム例]

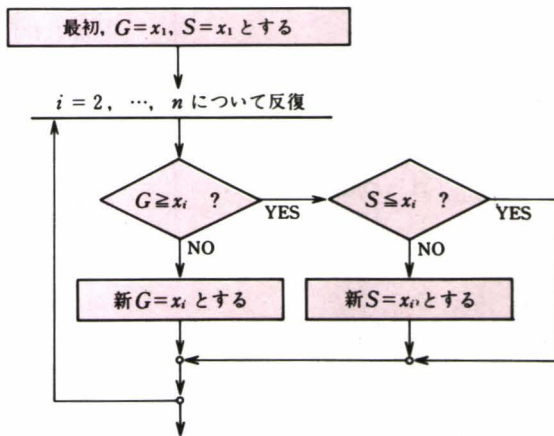
```
 10 INPUT N
 20 DIM A(N)
 30 FŌR I=1 TŌ N
 40     INPUT A(I)
 50 NEXT I
 60 S=0
 70 FŌR I=1 TŌ N
 80     S=A(I)+S
 90 NEXT I
100 PRINT "ゴウケイ ";S
```

あるいは, 次のようにしてもよい.

```
 10 INPUT N
 20 S=0
 30 FŌR I=1 TŌ N
 40     INPUT A
 50     S=A+S
 60 NEXT I
 70 PRINT "ゴウケイ ";S
```

4.13 最大値，最小値

■ 一組の値 x_1, x_2…, x_n の最大値 G を求めるには，次のようにすればよい．

（反復終了時の G の値が最大値である）

■ 最小値も同様な要領で求めることができる．

■ 最大値 G と最小値 S を同時に求めるには次のようにする．

（反復終了時の G，S の値が最大値，最小値である）

[プログラム例] データー a_1，a_2，…，a_n の最大値を求めるには次のようにすればよい．

```
300 MAX=A(1)
310 FŌR I=2 TŌ N
320   IF MAX<A(1) THEN MAX=A(I)
330 NEXT I
```

最大値だけでなく，その番号も必要な場合は次のようにする．

```
300 MAX=A(1) : L=1
310 FŌR I=2 TŌ N
320   IF MAX<A(I) THEN MAX=A(I):L=I
330 NEXT I
```

最小値を求めるには次のようにする．

```
340 MIN=A(1)
350 FŌR I=2 TŌ N
360   IF MIN>A(I) THEN MIN=A(I)
370 NEXT I
```

最大値と最小値を同時に求めるには次のようにする．

```
300 MAX=A(1)
310 MIN=A(1)
320 FŌR I=2 TŌ N
330   IF MAX<A(I) THEN MAX=A(I) :
                     (前行の続き)  GŌTŌ 350
340   IF MIN>A(I) THEN MIN=A(I)
350 NEXT I
```

絶対値の最大値を求めるなら次のようにする．

```
300 MAX=ABS(A(1))
310 FŌR I=2 TŌ N
320   IF MAX<ABS(A(I)) THEN MAX=ABS(A(I))
330 NEXT I
```

4.14 SWAP

これは二つの変数の内容を交換する命令で，制御文ではないが，制御文と一緒によく用いられる．SWAP文は次の形で書く．

SWAP　変数名，変数名

（例）　SWAP A，B

（例）　手もとに本機があったら，次のプログラムを実行させてみるとよい．

```
10 INPUT "a=";A
20 INPUT "b=";B
30 SWAP A,B
40 PRINT "a=";A
50 PRINT "b=";B
```

内容の入れ替わっているのがわかるであろう．

（実行例）

```
run
a=? 2
b=? 3
a= 3
b= 2
Ok
```

（使用例と実行例）

```
10 INPUT "a=";A
20 INPUT "b=";B
30 IF A<B THEN SWAP A,B
40 PRINT "ｵｵｷｲ ﾎｳﾊ";A
50 PRINT "ﾁｲｻｲ ﾎｳﾊ";B
```

```
run
a=? 298
b=? 808
ｵｵｷｲ ﾎｳﾊ 808
ﾁｲｻｲ ﾎｳﾊ 298
Ok
```

4.15 大きさの順に並べる

一組のデーター x_1, x_2, …, x_n を，大きさの順(大きい順，または小さい順）に並べ替えたいことがよくある．このような処理を**ソート（整列，sorting)** という．それには各種の方法があるが，最も簡単なのは交換法である．

交換法　これの基本形は「隣り合う二つのデーターを比較し，

正順（望ましい順序）になっていればそのまま

逆順（逆の順序）になっていれば入れ替える

という処理を，列の最初から最後まで何回もくりかえす」という方法である．詳しく検討してみると，反復回数は最大 $n-1$ 回で十分であり，第 i 回目には「列の最初から $n-i$ 番目まで」調べれば十分であることがわかる．

[プログラム例]

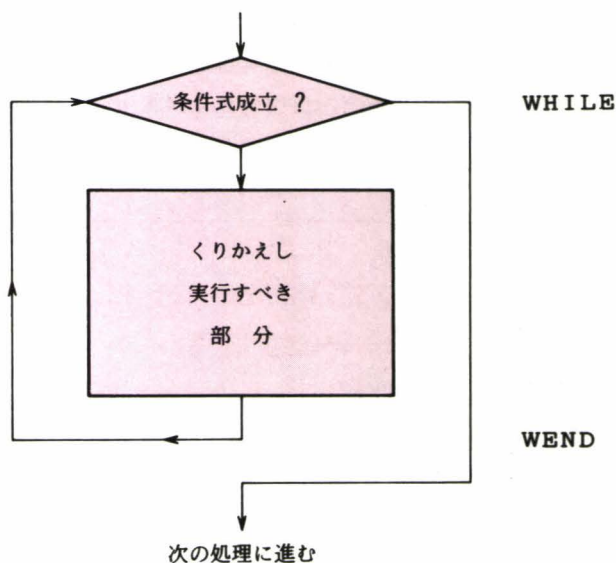
```
1 REM --- オオキイ ジュン ニ ナラベカエル プログラム ---
10 INPUT "N=",N
20 DIM A(N)
30 FOR I=1 TO N
40    PRINT "A(";I;")="; : INPUT A(I)
50 NEXT I
60 FOR M=N-1 TO 1 STEP -1
70    FOR J=1 TO M
80       IF A(J)<A(J+1) THEN SWAP A(J),A(J+1)
90    NEXT J
100 NEXT M
110 FOR I=1 TO N
120    PRINT A(I)
130 NEXT I
140 END
```

```
RUN
N=5
A( 1 )=? 9
A( 2 )=? 4
A( 3 )=? 7
A( 4 )=? 2
A( 5 )=? 5
 9
 7
 5
 4
 2
Ok
```

4. 16 WHILE～WEND

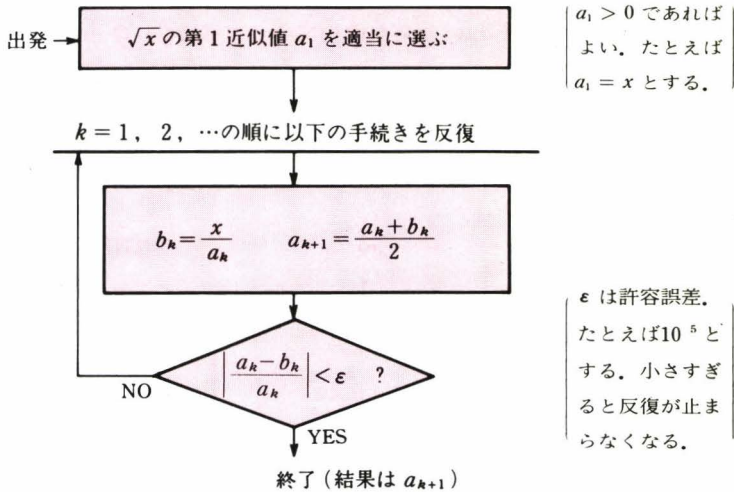
指定した条件が満たされている間だけ反復を行ないたい場合のためにWHILE，WENDという文を使用できる．書き方は

```
WHILE　条件式
    くりかえし実行すべき部分
WEND
```

概念図

[備考]　1）条件式はIF文の場合と同様，等式，不等式，またはそれらを論理記号でつないだ形で書く．

2）WHILE～WENDを入れ子の形で多重に用いることができる．その対応関係は「内側から順に」という形で処理される．

3）最初から条件式不成立の場合は，「くりかえし実行すべき部分」を1回も実行せずに先に進む．

$\sqrt{x}$ の値を求めるには組込み関数SQRを用いてもよいが，次のような逐次近似法で計算することもできる．このプログラムを作れ．

[解答]

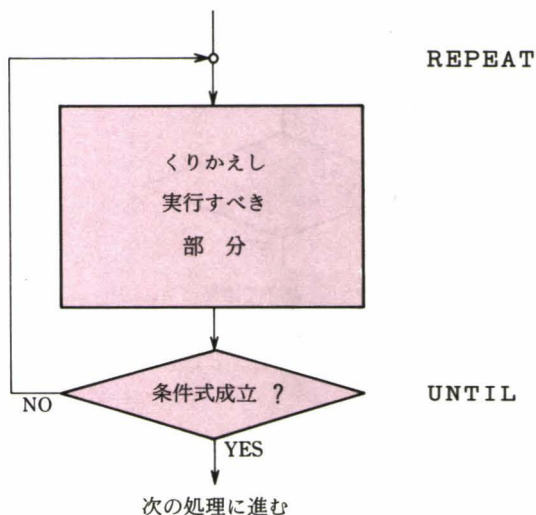
```
10 EPS=.000001
20 INPUT "X=",X
30 A=1
40 B=X/A
50 PRINT "A","B"
60 WHILE ABS((A-B)/A)>EPS
70    A=(A+B)/2
80    B=X/A
90    PRINT A,B
100 WEND
```

```
RUN
X=2
A             B
 1.5           1.3333333
 1.4166667                  1.4117647
 1.4142157                  1.4142114
 1.4142136                  1.4142136
OK
```

4.17 REPEAT〜UNTIL

指定した条件が満たされるまで反復を行なう場合のために**REPEAT，UNTIL**という文を使用できる．書き方は，

```
REPEAT
    くりかえし実行すべき部分
UNTIL  条件式
```

概念図

（例）

```
10 INPUT X
20 A=1 : B=X
30 REPEAT
40     B=(A+B)/2
50     A=X/B
60 UNTIL ABS(A-B)<1E-06
70 PRINT "root(X)=";A
```

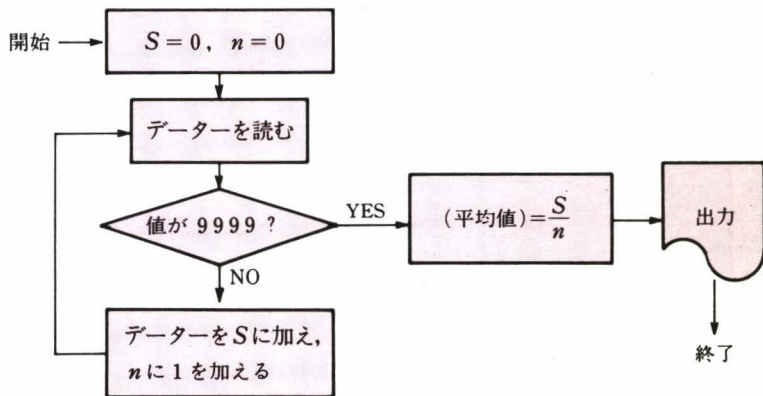
```
RUN
? 2
root(X)= 1.4142136
OK
```

［解説］ **WHILE**文だとループに入った時点で条件式の値が確定していなければならないのに対し，**REPEAT**だとループに入ってから値を決めればよいので使い易い．

例題 8 ── 不定箇数のデーターの読み込み

一組のデーターa_1, a_2, …, a_n を入力するとき，データーの箇数 n がわかっていれば，n を最初に入力することができるが，場合によっては，n がわからないとか，数えるのがめんどうだから数えないで済ませたいということがある，そのような場合には，最後のデーターに何か印を付けるか，あるいは最後のデーターの次に何か特別な値を入れるかして，入力を終了させればよい．

上記のような方式で，最後のデーターの次に9999を入れることにして，不定箇数のデーターを読み込み，その平均値を計算するプログラムを作れ．

[解答]

```
10 S=0 : N=0
20 INPUT A
30 REPEAT
40     S=S+A
50     N=N+1
60     INPUT A
70 UNTIL A=9999
80 PRINT "ﾍｲｷﾝﾁ ";S/N
90 END
```

[注意] 最後を表す印はパソコンで正確に（丸め誤差なしで）表せる数にしておかないと止まらなくなることがある．したがって小数部の付いた数や32768以上の整数は使わない方がよい．

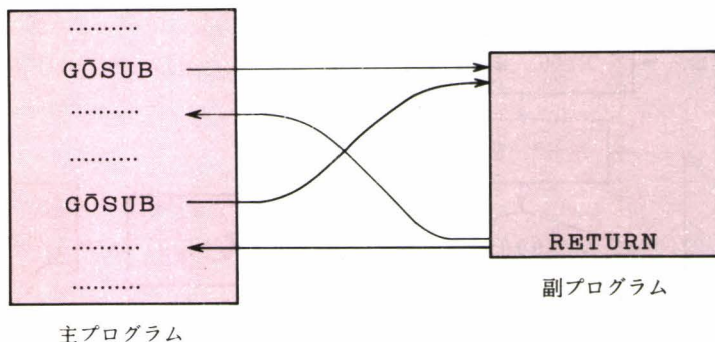
4.18 GOSUB～RETURN

これは，行って処理を行なったあと，もとの所にもどってきて続きを実行したい場合に用いる文で次のように書く*.

GŌSUB 行先の行番号

この文を実行すると，行番号で指定された所へ飛び，そこのプログラムを実行し，

RETURN

という文に達すると，もとの所(正確にいえばGŌSUB文の次の文)にもどる.

[備考] 他のプログラミング言語（たとえばFORTRANやPASCAL）と違って，BASICの副プログラムは変数，行番号などが主プログラムと全部共通であり，引数を書くことができない.

---

* SUBはsubroutineの意.

例題 9 ―― 時分秒の計算

「h_1 時 m_1 分 s_1 秒から h_2 時 m_2 分 s_2 秒までは何時間何分何秒か？」という計算（時分秒の引き算）をするサブルーチンを作れ．

[解答] 時，分，秒の単位を併用して時刻を表すことは日常よく行なわれている．その加減算を行なうには

- 60進法の加減算のプログラムを作る
- すべてを秒単位に換算して加減法を行なう

の二つの方法がある．IF 文の練習や n 進法演算の扱いに慣れるためには前者もよいが，実務には後者の方が簡単でよい．

[プログラム例]

```
1000 LABEL "HMS"
1010 T1=3600*H1+60*M1+S1
1020 T2=3600*H2+60*M2+S2
1030 T=T2-T1
1040 H=T ¥ 3600
1050 R=T MOD 3600
1060 M=R ¥ 60
1070 S=M MOD 60
1080 RETURN
```

[メイン・プログラムの例]

```
10 INPUT "時,分,秒(1)=",H1,M1,S1
20 INPUT "時,分,秒(2)=",H2,M2,S2
30 GOSUB "HMS"
40 PRINT H;"時カン";M;"分";S;"秒"
50 END
```

```
RUN
時,分,秒(1)=3,45,12
時,分,秒(2)=7,21,36
 3 時カン 36 分 36 秒
OK
```

「時」「分」「秒」は GRAPH ＋ 8 ，GRAPH ＋ 9 ，GRAPH ＋ 0 で入力する．

[注意] RUN ⏎ で開始すると最も若い行番号の文から実行されるから，上の例のように，メイン・プログラムは前（若い番号），サブルーチンは後（あとの番号）に置くのが普通である．その場合，メイン・プログラムの最後のEND文は省略できない（そうしないとメイン・プログラムの後了後，サブルーチンの中に飛び込んでしまう）．

例題 10　日数計算

西暦 y_1 年 m_1 月 d_1 日から，西暦 y_2 年 m_2 月 d_2 日までの日数を計算するプログラムを作れ．ただし，$y_1 \geqq 1600$ とする．

[解答] 年，月，日を別個に引算して，引けなければ上位の値から借りてくる，という方法も考えられるが，月の日数が一定でないため，かなり複雑なプログラムになる．それよりも，一定の基点（たとえば，1600年1月0日）からの日数を計算するプログラムを作り，その日数で引き算を行なう方が簡単である．

まず，1月0日からm月d日までの日数を計算する方法を考えてみるとよい．それには，各月の0日から何日目になるかを電卓で計算して表を作り記憶させておくと簡単である．

1月	2月	3月	4月	5月	6月
0	31	59	90	120	151
7月	8月	9月	10月	11月	12月
181	212	243	274	304	334

これは平年の場合である．うるう年には，3月以降，上記の値に1を加える．上記の値にdを加えれば，1月0日からm月d日までの日数**N1**がわかる．

次に，1600年1月0日から，y 年1月0日までの日数**N2**を計算する．それにはいろいろな方法が考えられるが，最も簡単かつ明快なのは，ひとまず1年を365日として計算し，それにうるう年のぶんの補正を加える，という方法である．1年を365日として計算すると，**N2**の概算値は $365\times(y-1600)$ となる．これに「1600年から y 年までのうるう年の回数」を加えればよい．その回数は組込み関数**INT**を用いて　**INT((Y−1600)/4)+1**　という式で求めることができる．この式は「西暦を4で割り切れる年はうるう年である」という規則に基づくものである．ところで，暦法には100で割り切れる年は平年とする，ただし400で割り切れる年はうるう年とする，という規則もある．この補正を行うには，

INT((Y−1600)/100)+1　を引き，

INT((Y−1600)/400)+1　を加える

とすればよい．以上をまとめると，プログラムは次のようになる．

```
10 REM ------- 年月日 -------
20 DIM C(12)
30 C(1)=0:C(2)=31:C(3)=59
40 C(4)=90:C(5)=120:C(6)=151
50 C(7)=181:C(8)=212:C(9)=243
60 C(10)=273:C(11)=304:C(12)=334
70 INPUT "Y1,M1,D1=";Y1,M1,D1
80 INPUT "Y2,M2,D2=";Y2,M2,D2
90 Y=Y1 : M=M1 : D=D1
100 GOSUB 210
110 A1=N
120 Y=Y2 : M=M2 : D=D2
130 GOSUB 210
140 A2=N
150 A=A2-A1
160 PRINT
170 PRINT Y1;"年 ";M1;"月 ";D1;"日 ｶﾗ"
180 PRINT Y2;"年 ";M2;"月 ";D2;"日 ﾏﾃﾞﾊ"
190 PRINT A;"日 ﾃﾞｽ｡"
200 END
210 REM ------- ｶﾝｻﾝ -------
220 Z=Y-1600
230 N2=365*Z+INT(Z/4)-INT(Z/100)+INT(Z/400)
240 IF Z>0 THEN N2=N2+1
250 N1=C(M)+D
260 N=N1+N2
270 IF (Y MOD 4)<>0 THEN RETURN
280 IF (Y MOD 400)=0 GOTO 300
290 IF (Y MOD 100)=0 THEN RETURN
300 REM ------- ｳﾙｳﾄﾞｼ -------
310 IF M>2 THEN N=N+1
320 IF Z>0 THEN N=N-1
330 RETURN
```

例題 11 **曜日**

西暦 y 年 m 月 d 日が何曜日かを調べるプログラムを作れ．

[解答] 例題10で説明した方法で「1600年1月0日からの日数」N を計算し，それを7で割った余りを R とすれば，次のようになる．

R	0	1	2	3	4	5	6
曜日	金	土	日	月	火	水	木

```
10 REM ------- ヨウビ" -------
20 DIM C(12),W$(6)
30 C(1)=0:C(2)=31:C(3)=59
40 C(4)=90:C(5)=120:C(6)=151
50 C(7)=181:C(8)=212:C(9)=243
60 C(10)=273:C(11)=304:C(12)=334
70 W$(0)="キン":W$(1)="ド"":W$(2)="日"
80 W$(3)="月":W$(4)="カ":W$(5)="スイ"
90 W$(6)="モク"
100 INPUT "年,月,日 = ";Y,M,D
110 GOSUB 170
120 R=N-INT(N/7)*7
130 PRINT
140 PRINT Y;"年";M;"月";D;"日 ハ ";
150 PRINT W$(R);" ヨウビ" デ"ス。"
160 END
170 REM --- 1600.1.0 カラノ ニッスウニ カンサン ---
180 Z=Y-1600
190 N2=365*Z+Z¥4-Z¥100+Z¥400
200 IF Z>0 THEN N2=N2+1
210 N1=C(M)+D
220 N=N1+N2
230 IF (Y MOD 4)<>0 THEN RETURN
240 IF (Y MOD 400)=0 GOTO 260
250 IF (Y MOD 100)=0 THEN RETURN
260 IF M>2 THEN N=N+1
270 IF Z>0 THEN N=N-1
280 RETURN
```

```
RUN
年,月,日 = ? 1984,1,1
 1984 年 1 月 1 日 ハ 日 ヨウビ" デ"ス。
```

[備考] 行20，70～90，150に現れる変数**W$**は，次の章で説明する文字列型の変数である．詳しくは5.2節参照．

例題 12 ―― カレンダー

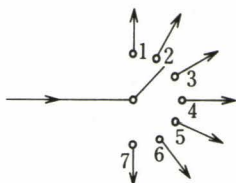
y と m を読み込み（ただし $y \geqq 1600$ とする）西暦 y 年 m 月のカレンダーを出力するプログラムを作れ．

[解答] 西暦 y 年 m 月 1 日が何曜日になるかは前問の要領で計算できるから，その曜日の位置から始めて，カレンダーの形式で数字 1 から31まで（小の月は30，29または28まで）を出力すればよい．カレンダー形式にまとめる方法はいろいろ考えられるが，以下に示すのは比較的簡単な方法の一例である．このプログラムではカラー・ディスプレイを用いて，日曜日は赤，土曜日は青，その他の曜日は白で表示するようにしてみた．祝日を赤にするとか，祝日と日曜が重なった場合に翌日を赤にするとかいうような処理は省略してあるが，この例でだいたいの要領はわかると思うから，あとは各自で工夫されたい．後述のグラフィック・ディスプレイの技法を活用して見ばえのするカレンダーを作ってみるとよい．

```
10 REM ------- カレンダー -------
20 INIT : CLS 4
30 COLOR 4
40 INPUT "年,月 ";Y,M
50 COLOR 6
60 PRINT : PRINT
70 PRINT SPC(12);Y;" 年";M;"月"
80 PRINT
90 DIM C(12),D(12),W$(7)
100 D(1)=31 : D(2)=28 : D(3)=31
110 D(4)=30 : D(5)=31 : D(6)=30
120 D(7)=31 : D(8)=31 : D(9)=30
130 D(10)=31: D(11)=30: D(12)=31
140 IF (Y MOD 4)=0 AND (Y MOD 100)<>0 OR (Y MOD 400)=0
150 C(1)=0                       (前行の続き) THEN D(2)=29
160 FOR I=1 TO 11
170     C(I+1)=C(I)+D(I)
180 NEXT I
190 COLOR 2
200 PRINT "  日   ";
210 COLOR 7
220 PRINT "  月   ";
230 PRINT "  火   ";
240 PRINT "  水   ";
250 PRINT "  木   ";
260 PRINT "  金   ";
270 COLOR 1
280 PRINT "  土   "
```

（→166ページに続く）

4.19 ON～GOTO, ON～GOSUB

一度にたくさんの枝に分岐したいことがある．

番号をキー（選択の基準）としてこのような枝分かれを行なうには，

ŌN　番号　GŌTŌ　行先$_1$，行先$_2$，…，行先$_n$

という文を用いると便利である．これは

番号が 1 ならば行先$_1$ に行け

2　　　行先$_2$

⋮　　　⋮

n　　　行先$_n$

ということを表している．行先は行番号またはラベルで指定する．

（例）　ŌN K GŌTŌ 100,300,1750

番号の所には式を書いてもよい．

（例）　ŌN N−40 GŌTŌ 200,280,170

式の値が整数にならなかった場合には小数部分は切り捨てられる．その結果が1～n 以外になった場合（いいかえれば，該当する行先が無い場合）は次の文に進む．

同様に

ŌN　番号　GŌSUB　行先$_1$，行先$_2$，…，行先$_n$

という書き方もできる．この場合は，番号で選ばれた行先に進み，そこのプログラムを実行し，**RETURN**文に到達するとそこからもとの所（**ŌN**文の次の文）にもどってくる．

例題 13 スイッチング

面積や体積の計算をするための汎用プログラムを作りたい．まず機能選択コードKを読み込み，Kが1ならば三角形の面積，2ならば円の面積，3ならば楕円の面積，4ならば球の面積，5ならば円錐の体積を計算するプログラムに進み，必要なデーターを読み込んで計算を行なうようにせよ．

[解答]

```
10 REM --- メンセキ,タイセキ ---
20 PRINT "1 サンカクケイ"
30 PRINT "2 エン"
40 PRINT "3 ダエン"
50 PRINT "4 キュウ"
60 PRINT "5 エンスイ"
70 INPUT "ドレニシマスカ?",K
80 ON K GOSUB "サンカクケイ","エン","ダエン","キュウ","エンスイ"
90 END
100 LABEL "サンカクケイ"
110 INPUT "テイヘン";A
120 INPUT "タカサ";H
130 PRINT "メンセキハ ";A*H/2
140 RETURN 10
150 LABEL "エン"
160 INPUT "ハンケイ";R
170 PRINT "メンセキハ ";π*R^2
180 RETURN 10
190 LABEL "ダエン"
200 INPUT "チョウケイ";A
210 INPUT "タンケイ";B
220 PRINT "メンセキハ ";π*A*B
230 RETURN 10
240 LABEL "キュウ"
250 INPUT "ハンケイ";R
260 PRINT "タイセキハ ";(4/3)*π*R^3
270 RETURN 10
280 LABEL "エンスイ"
290 INPUT "ハンケイ";R
300 INPUT "タカサ";H
310 PRINT "タイセキハ ";(1/3)*H*π*R^2
320 RETURN 10
```

4.20 文 法 詳 説

コンピューターの内部では，すべての情報は 0 と 1 の組合せで表現され，処理されるが，BASIC のセンスでいえば「0 と 1 の組合せ」というのは要するに数値のことであるから「すべての情報は数値の形で表現され処理される」といってよい．

IF文や**WEILE**文には，たとえば

　　A=B　　とか　　**(A<B)AND(C>=D)**

というような「条件式」が現れる．このような条件式が，成立している（真である）とか，成立していない（偽である）というような情報も数値として表現される．本機の BASIC では

真（条件成立）	を	**−1**
偽（条件不成立）	を	**0**

で表している．

このように数値の形で表された論理情報は，変数の値として代入したり，演算したり，出力したりすることができる．

論理値の代入　　たとえば，二つの数値**A**，**B**を入力し，不等式

　　A>B

が成立するか否かを調べ，その結果（真，偽のいずれか）を変数**C**に代入し，出力してみよう．

```
10 INPUT "a,b=",A,B
20 C=A>B
30 PRINT C
```

```
RUN
a,b=2,3
 0
```

```
RUN
a,b=3,2
-1
```

論理値の入力　BASIC 以外のプログラミング言語 (FORTRAN など) には論理変数というのがあって，その入出力が可能であるが，BASIC でも「真は－1，偽は0で表す」という規則を知っていれば，論理値を入力することができる.

(例)

```
10 INPUT "a=",A
20 IF A THEN PRINT"T" ELSE PRINT"F"
```

```
RUN
a=-1
T
```

```
RUN
a=0
F
```

IF 算術式 THEN という文が使える　上のプログラム例からもわかるように，ＩＦ文の「条件式」という所に，普通の変数とか，一般には算術式を書いても，ちゃんと実行してくれる．その場合

> 変数（一般には算術式）の値が0でなければ「真」
> 変数（一般には算術式）の値が0ならば「偽」

として扱われる.

算術式の中に論理式を書ける　算術式（数値変数の式，要するに普通の式）の中の一つの項として論理式を書くことができる．項の値は，前述のとおり，真ならば－1，偽ならば0となる.

(例)

```
10 INPUT "A=",A
20 INPUT "B=",B
30 INPUT "C=",C
40 INPUT "D=",D
50 X=(A>B)+(C=D)
60 PRINT X
```

```
RUN
A=1
B=2
C=3
D=4
 0
```

```
RUN
A=3
B=2
C=1
D=1
-2
```

この性質は，たとえば一群のデーターを調べて「特定の条件に合うものの箇数を数える」というような場合に応用できる.

例題 14 ―― **数える**

N 日間の，正午の気温の観測値 $T_1, T_2, \cdots, T_N$ を読み 30℃ 以上であった日数 K を数えてプログラムを作れ．

[解答]

```
10 INPUT "N=",N
20 FOR I=1 TO N
30   INPUT T
40   K=K-(T>=30)
50 NEXT I
60 PRINT "30ﾄﾞ ｲｼﾞｮｳ ﾉ ﾋ ﾊ";K;"日"
```

[解説]　30℃ 以上ならば（**T>=30**）が真だから，その値は**−1**となり，行40ではそれを**K**から引くから，**K**の値に**1**が加えられることになる．一方，30℃ 未満ならば（**T>=30**）が偽だから，その値は**0**となり，行40を実行しても**K**の値は変らない．

数値変数の論理演算ははビット毎の論理演算になる　演算子**AND**，**ŌR**，**NŌT**は，本来，論理値（真，偽）を対象とするものであるが，じつは，数値変数（一般には算術式）に対しても**AND**，**ŌR**，**NŌT**が使えるのであって，その場合「対応するビット毎の演算」になる．ビット毎の演算規則は次の通りである．

(0 ŌR 0)=0	(0 AND 0)=0
(0 ŌR 1)=1	(0 AND 1)=0
(1 ŌR 0)=1	(1 AND 0)=0
(1 ŌR 1)=1	(1 AND 1)=1
(NŌT 0)=1	(NŌT 1)=0

（例）　3 を 2 進法で表すと　　**011**

　　　5 を 2 進法で表すと　　**101**

　　　ビット毎の**ŌR**をとると　　**111**

　　　ビット毎の**AND**をとると　　**001**

であるから，結果を10進法で読むと次のようになる．

　　（3 ŌR 5）　の値は　7

　　（3 AND 5）　の値は　1

```
10 INPUT "A=",A
20 INPUT "B=",B
30 C=A AND B
40 D=A OR B
50 E=NOT A
60 PRINT "A AND B",C
70 PRINT "A OR B",D
80 PRINT "NOT A",E
```

```
RUN
A=3
B=5
A AND B        1
A OR B         7
NOT A         -4
```

```
RUN
A=255
B=257
A AND B        1
A OR B       511
NOT A       -256
```

[応用] ビット毎の演算は，特定の桁を残して他の部分を消す操作(「マスクをかける」という）に応用することができる．たとえば，**N**の値が奇数であるか否かは，**IF(N AND 1)**で判定できる．ただし，このような扱いができるのは－32768以上，＋32767以下の整数に限られる．

```
10 INPUT "N=",N
20 IF (N AND 1) THEN PRINT "ｷｽｳ" ELSE PRINT "ｸﾞｳｽｳ"
30 GOTO 10
```

```
RUN
N=5
ｷｽｳ
N=6
ｸﾞｳｽｳ
N=7
ｷｽｳ
```

XOR, IMP, EQV　　論理演算子には，**AND**，**ŌR**，**NŌT**のほかに，**XŌR**，**IMP**　**EQV**がある．

A	B	(A XŌR B)	(A IMP B)	(A EQV B)
0	0	0	1	1
0	1	1	1	0
1	0	1	0	0
1	1	0	1	1

[10進→2進変換] ビット単位の演算を扱う際に,数値のビット・パターンを知りたくなることがよくある. そのための, 10進→2進変換のプログラムを作ってみよう. 作り方はいろいろある. コンピューターの内部では, 数値は既に2進法に変換されているのであるから, 前述の論理演算機能を利用してビット・パターンを直接調べる, というのも一案であるが, 以下に示すのは, もっと平凡なプログラム例で,

整数 A を入力する

それを2で割って*, 商を C とし, 余りを b_0 とする

C を2で割って*, 商を新 C とし, 余りを b_1 とする

新 C を2で割って*, 商を新 C とし, 余りを b_2 とする

………以下同様………

上で得られた b_0, b_1, b_2, …を逆順に出力する**

という方法を用いている.(**PRINT USING**に関しては6.6節参照)

```
10 INPUT "A=",A
20 DIM B(31)
30 FOR I=0 TO 31
40    C=INT(A/2)
50    B(I)=A-2*C
60    A=C
70 NEXT I
80 FOR I=31 TO 0 STEP -1
90    PRINT USING "#",B(I);
100 NEXT I
110 PRINT
120 END
```

```
RUN
A=5
00000000000000000000000000000101
```

[別解] 本機の場合, 関数**BIN$**を用いて

```
PRINT BIN$(A)
```

としてもよい.

---

* 整除, すなわち商が整数になる範囲で割れるところまで割る.

** 最下位が b_0, その上の桁が b_1, …となるから, 16桁なら

b_{15} b_{14} … b_2 b_1 b_0

の形で出力すればよい.

5 文字列処理

5.1 文字列型

BASICではデーターとして文字列を扱うことができる．文字列というのは，文字を1列に並べたもので，たとえば，

単独の文字	(例)	A
文字の組合せ	(例)	CZ−800C
単　語	(例)	パソコンテレビ
文	(例)	Let it be.
名　前	(例)	サトウ　サンペイ
数　式	(例)	a＋b
プログラム	(例)	10 INPUT A

などをデーターとして扱うことができるわけである．このようなデーターに関し，

入力，出力

代　入

一部分を取り出す

二つの文字列をつないで一つの文字列にする

比　較

などの処理ができる．

長さの制限　取り扱うことのできる文字列の長さは，0字から255字までである．

使用できる文字　アルファベットの大文字，小文字，カナ文字，数字，記号など，要するにキーボードから入力できるものは何でもよい．また一部の制御コード（たとえば ⏎ TAB など）も扱うことができる．詳しくは付録Bの表を参照のこと．

5.2 変数および定数

文字列型変数名　$ という属性文字を付けて表す．その他の規則は普通の変数名の規則に準ずる．

(例)　A$
　　　NAMAE$

型宣言　いちいち $ を付けるのがめんどうならば，型宣言

　　DEFSTR　頭文字

または

　　DEFSTR　頭文字の範囲

を用いてもよい．DEF は define，STR は string の意味である．

(例)　DEFSTR　A
　　　DEFSTR　P−R

文字列型定数　2重引用符 " で囲んで表す．

(例)　"A"　"="　"♡"　"〒"　"π"
　　　"Jacques Loussier"
　　　"(A+B)/C"

文字列型配列　文字列型についても，配列を用いることができる．宣言は

　　DIM　配列名(寸法)

により行なう．

(例)　DIM A$(100),C$(30),M$(7,5,3)

5.3　代入および比較

文字列型変数の代入　代入の書き方は文字列型変数でも同じである.

（例）　**A$=B$**

　　　MŌJI$="A"

代入先の変数の文字列長さは，式の方で定まる文字列の長さに合わせて設定（必要ならば変更）される.

（例）　**A$**の内容が**ABC**のとき，代入文

　　　A$="PQRST"

を実行すると，以前の内容**ABC**は消され，かわりに**PQRST**が入る.文字数は3字から5字になるが，このような代入も可能である.

比　較　二つの文字列が等しいか否かを**IF**文で判定することができる.

（例）　**IF　A$=B$　THEN　PRINT　"ヒトシイ"**

　　　IF　I$="Y"　GŌTŌ　300

IF文によって文字列の大小を比較して分岐することができる. その際の大小関係はASCIIコード*の大小に従う.

比較する文字列は2字以上でもよい. その場合，次のように扱う.

　　先頭から順に比較する

　　文字列の終端より先は値ゼロとみなす

（例）　**A＜B＜C＜…＜Z＜ア＜イ＜ウ＜…＜ワ＜ヲ＜ン**

　　　A＜AA＜AA空白**＜AB**

---

*　付録B参照

5.4 入出力（基本）

文字列型データーの入力　普通はINPUT文を用いる.

（例）　INPUT　A$

プログラムで字数を指定する必要はない. また, 普通は, 入力文字列を"印で囲む必要もない. その場合, 入力要求（?印）が出てから, 入力データーの区切りを示す記号（コンマまたは⏎）の直前までが一つの文字列となる. ただし"印で囲んでおけばコンマを含む文字列を入力することができる.

文字列型データーの出力　ディスプレイに表示するにはPRINT文を用いる.

（例）　PRINT　A$

プリンターに出力するにはLPRINT文を用いる.

（例）　LPRINT　S$

例題 1　会計

品名, 単価, 数量を読み込み, 金額を計算し, プリンターに出力するプログラムを作れ.

[解答]　品名を文字列型変数HIMMEI$に読み込み, プリンターに出力する. 単価には変数名TANKAを使いたいが, 予約語の関係で文法違反になるので, 代りにTAMKAとする. 金額を計算したついでに, 合計金額も計算することにしよう. 合計としては, 小計と総計の2種類を区別し, 品名の入力時に

*印を入れたら, 小計を印刷して小計をクリア

#印を入れたら, 総計を印刷して計算終了

ということになる.

行180, 200, 280, 300に用いられているSTRING$という関数は, 同じ文字を何字も続けて書くかわりに　STRING$（字数, "文字"）で表す書き方で, 詳しくは6.8節で説明する. なお, ここでの－印および＝印はグラフィック記号の「横けい」,「2重横けい」である.

```
10 REM ------- ｶｲｹｲ -------
20 SOUKEI=0
30 REM --- major loop ---
40 SYOUKEI=0
50 LPRINT "ﾋﾝﾒｲ","ﾀﾝｶ","ｽｳﾘｮｳ","ｷﾝｶﾞｸ"
60 REM --- minor loop ---
70 INPUT "ﾋﾝﾒｲ";HIMMEI$
80 IF HIMMEI$="#" GOTO "SUBTOTAL"
90 IF HIMMEI$="*" GOTO "SUBTOTAL"
100 INPUT "ﾀﾝｶ";TAMKA
110 INPUT "ｽｳﾘｮｳ";SUURYOU
120 KINGAKU=TAMKA*SUURYOU
130 PRINT "ｷﾝｶﾞｸ ﾊ ";KINGAKU;"円ﾃﾞｽ｡"
140 LPRINT HIMMEI$,TAMKA,SUURYOU,KINGAKU
150 SYOUKEI=SYOUKEI+KINGAKU
160 GOTO 60
170 LABEL "SUBTOTAL" : REM --- "*","#" ---
180 PRINT STRING$(20,"—")
190 PRINT "ｺﾞｳｹｲ",SYOUKEI
200 LPRINT STRING$(32,"—")
210 LPRINT "ｺﾞｳｹｲ",,,SYOUKEI
220 LPRINT : LPRINT
230 SOUKEI=SOUKEI+SYOUKEI
240 IF HIMMEI$="#" GOTO "GRANDTOTAL"
250 PAUSE 50 : CLS
260 GOTO 30
270 LABEL "GRANDTOTAL" : REM --- "#" ---
280 PRINT STRING$(20,"_")
290 PRINT "ｿｳｹｲ",SOUKEI
300 LPRINT STRING$(32,"_")
310 LPRINT "ｿｳｹｲ",,,SOUKEI
320 END
```

［出力例］

```
ﾋﾝﾒｲ        ﾀﾝｶ       ｽｳﾘｮｳ      ｷﾝｶﾞｸ
ｺｰﾋｰ         350       2          700
ｱｲｽｸﾘｰﾑ      330       1          330
------------------------------------------
ｺﾞｳｹｲ                             1030

ﾋﾝﾒｲ        ﾀﾝｶ       ｽｳﾘｮｳ      ｷﾝｶﾞｸ
ﾋﾟﾗﾌ         500       1          500
ｺｰﾋｰ         350       1          350
------------------------------------------
ｺﾞｳｹｲ                             850

ﾋﾝﾒｲ        ﾀﾝｶ       ｽｳﾘｮｳ      ｷﾝｶﾞｸ
ｺｰﾗ          280       2          560
ｹｰｷ          300       1          300
------------------------------------------
ｺﾞｳｹｲ                             860

__________________________________________
ｿｳｹｲ                              2740
```

［備考］ 左の出力例を見ると，数値が左づめになっていて気持が悪いであろうが，後述の**PRINT USING**というのを使えば，右づめに出力することができる（6.6節参照）．

例題 2　HAPPY BIRTHDAY

名前を入力し，それを HAPPY BIRTHDAY の歌に組み込んで，

HAPPY BIRTHDAY TŌ YŌU.
HAPPY BIRTHDAY TŌ YŌU.
HAPPY BIRTHDAY DEAR 名前 SAN.
HAPPY BIRTHDAY TŌ YŌU.

と出力するプログラムを作れ．

[解答]　このプログラムには，いろいろな書き方がある．下記はその一例であるが，これにこだわらずに各自で工夫するとよい．

```
 10 HB$="HAPPY BIRTHDAY "
 20 TY$="TŌ YŌU."
 30 PRINT "オナマエヲ イレテクダ゛サイ"
 40 INPUT NAME$
 50 PRINT                              1行アケル
 60 PRINT HB$;TY$
 70 PRINT HB$;TY$
 80 PRINT HB$;"DEAR ";NAME$;" SAN."
 90 PRINT HB$;TY$
100 PRINT:PRINT:PRINT                  3行アケル
110 GŌTŌ 30
```

[出力例]

```
オナマエヲ イレテクダ゛サイ
? キヨシ

HAPPY BIRTHDAY TO YOU.
HAPPY BIRTHDAY TO YOU.
HAPPY BIRTHDAY DEAR キヨシ SAN.
HAPPY BIRTHDAY TO YOU.



オナマエヲ イレテクダ゛サイ
? タケシ

HAPPY BIRTHDAY TO YOU.
HAPPY BIRTHDAY TO YOU.
HAPPY BIRTHDAY DEAR タケシ SAN.
HAPPY BIRTHDAY TO YOU.
```

例題 3 五十音順に並べかえる

一組(全部でN箇)の地名(または人名,駅名などでもよい)を読み込み,それを五十音順に並べかえて出力するプログラムを作れ.

[解答] 並べかえの方法として,4.14節で述べた交換法を用いるとすればプログラムは次のようになる.

```
10 REM ------- input -------
20 INPUT "N=",N
30 DIM A$(N)
40 FOR I=1 TO N
50    INPUT A$(I)
60 NEXT I
70 REM ------- sort -------
80 FOR MADE=N-1 TO 1 STEP -1
90    FOR I=1 TO MADE
100        IF A$(I)>A$(I+1) THEN SWAP A$(I),A$(I+1)
110    NEXT I
120 NEXT MADE
130 REM ------ output -------
140 FOR I=1 TO N
150    PRINT A$(I)
160 NEXT I
```

[実行例]

```
run
N=8
? ｼﾌﾞﾔ
? ﾊﾗｼﾞｭｸ
? ﾖﾖｷﾞ
? ｼﾝｼﾞｭｸ
? ｼﾝｵｵｸﾎﾞ
? ﾀｶﾀﾉﾊﾞﾊﾞ
? ﾒｼﾞﾛ
? ｲｹﾌﾞｸﾛ
ｲｹﾌﾞｸﾛ
ｼﾌﾞﾔ
ｼﾝｵｵｸﾎﾞ
ｼﾝｼﾞｭｸ
ﾀｶﾀﾉﾊﾞﾊﾞ
ﾊﾗｼﾞｭｸ
ﾒｼﾞﾛ
ﾖﾖｷﾞ
Ok
```

5.5 文字列型の演算

文字列型データーに関し次のような処理が可能である.

1) **二つの文字列をつないで一つの文字列にする**　これは＋印で表す.

(例) A$の内容がCAN
　B$の内容がŌN
ならば,
　A$+B$はCANŌN

2) **左から*n*字をとり出す**　これは関数で,次のように表す.
　LEFT$(文字列型変数名,*n*)

(例) 人名が変数NAMAE$に入っているとき,その頭文字をとり出すには,
　LEFT$(NAMAE$,1)
とすればよい.

3) **右から*n*字をとり出す**　これは関数で,次のように表す.
　RIGHT$(文字列型変数名,*n*)

(例) I$の内容がタノキンならばRIGHT$(I$,2)の値はキンになる.

4) **文字列の中央部分(左から*m*の所から*n*字)をとり出す**
これは関数で,次のように表す.
　MID$(文字列型変数名,*m*,*n*)

(例) MŌJI$の内容がCANDYのとき,
　MID$(MŌJI$,2,3)
の値はANDになる.

5) **文字列の変更**　左から*m*字めの所から*n*字変更したい場合,
　MID$(文字列型変数名,*m*,*n*)="新しい文字列"
と書く(右辺は文字列型変数名や文字列式でもよい).

6) **文字列の中に，指定された綴りが含まれているか否かを調べ，含まれている場合は，その位置（綴りの先頭が文字列の左から何字目にあるか）を調べる**　これは，普通，

INSTR(文字列型変数名,"綴り")

の形で書く*(これには$が付かないことに注意!)．その値は

その文字列にその綴りが含まれない場合には0

含まれる場合はその位置（左からの字数で表した先頭位置）

になる．なお，照合開始位置を指定することもできる．その場合は次のように書く．

INSTR(照合開始位置,文字列型変数名,"綴り")

(例)　C$の内容が

```
10 IF X=Y GŌTŌ 700
↑── 先頭     ↑──── 11字目
```

のとき，

INSTR(C$,"GŌTŌ")

の値は11になる．

7) **文字列の長さを調べる**　これは関数で，

LEN(文字列型変数名)

と書く．

(例)　S$の内容が，**ジ゛ュケ゛ムジ゛ュケ゛ムコ゛コウノスリキレカイシ゛ャリスイキ゛ョノスイキ゛ョウマツウンキ゛ョウマツフウライマツクウネルトコロニスムトコロヤフ゛ラコウシ゛ノフ゛ラコウシ゛ハ゜イホ゜ハ゜イホ゜ハ゜イホ゜ノシューリンカ゛ンシューリンカ゛ンノク゛ーリンタ゛イク゛ーリンタ゛イノホ゜ンホ゜コヒ゜ーノホ゜ンホ゜コナチョウキュウメイノキュウスケ**　ならばLEN(S$)の値は165になる．

---

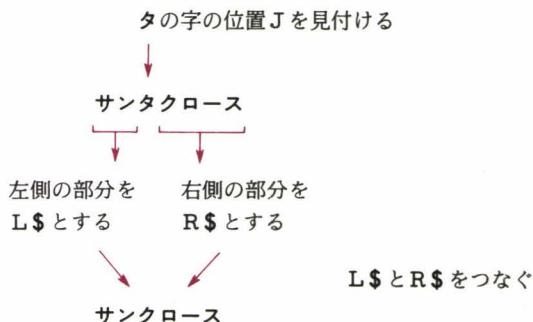
*　一般には，INSTR(文字列式,文字列式)の形が許される．第1引数が被検索文字列，第2引数が検索する綴りを表す．

例題 4　タヌキ言葉

文字列A$の中から「タ」の字を取り除くプログラムを作れ.

[解答]　タヌキ言葉というのは子供の遊びである. 文章から「タ」の字を全部抜いて話すのであるが, うっかり抜き忘れたり,「タ」の字を除くとおかしな感じになったりして面白い. 終戦直後はよく停電があり, 停電すると何もできないので, よくこんなことをして遊んだ.

BASICでこれを行なうには, まずINSTRで「タ」の字の位置をさがし, それより左の部分とそれより右の部分をLEFT$, RIGHT$で取り出し, 演算子＋で前後を結合すればよい.

(例)

したがって, プログラムの基本形は次のようになる.

```
10 INPUT A$
20 N=LEN(A$)
30 J=INSTR(A$,"タ")
40 L$=LEFT$(A$,J-1)
50 R$=RIGHT$(A$,N-J)
60 B$=L$+R$
70 PRINT B$
```

[実行例]

```
RUN
? イタチ
イチ
```

左記のプログラムは「タ」の字が1箇所だけならばよいが，二つ以上あると困る．その点を改良すると次のようになる．

```
10 INPUT A$
20 K=1
30 N=LEN(A$)
40 J=INSTR(K,A$,"タ")
50 IF J=0 GOTO 120
60 L$=LEFT$(A$,J-1)
70 R$=RIGHT$(A$,N-J)
80 A$=L$+R$
90 N=N-1
100 K=J
110 GOTO 40
120 PRINT A$
```

```
RUN
? タヌキ
ヌキ
```

```
RUN
? ハタハタ
ハハ
```

```
RUN
? カタタタキ
カキ
```

これでだいたいOKであるが，じつは「タ゛」の場合に困る．

(例)　メタ゛カ　→　メ゛カ

これを解決するには，「タ゛」の場合，濁点もいっしょに取り除くという方式と，外国語のようにtaとdaは別音と考えて「タ゛」は残すという方式とがある．後者のプログラム例を以下に示す．

```
10 INPUT A$
20 K=1
30 N=LEN(A$)
40 J=INSTR(K,A$,"タ")
50 IF J=0 GOTO 150
60 IF J=N GOTO 90
70 MIGI$=MID$(A$,J+1,1)
80 IF MIGI$="゛" THEN K=J+1 : GOTO 40
90 L$=LEFT$(A$,J-1)
100 R$=RIGHT$(A$,N-J)
110 A$=L$+R$
120 N=N-1
130 K=J
140 GOTO 40
150 PRINT A$
```

```
RUN
? メタ゛カ
メタ゛カ
```

これでもまだぐあいの悪いのが

　　ワンタン　　タンメン　　タップ゜タ゛ンス

などである．ひとつ各自で工夫されたい．

5.6　ASCII コード

本機の内部で，文字は0～225の番号で表されている．文字と番号の対応は，付録Bの表のようになっている．この符号系は，英字，数字，記号に関する限り，ASCII（アスキーと読む．アメリカの標準規格）の符号系と同一なので，ASCII コードと呼ばれている．

ASCII コードを文字に変換するには次のように書く．

CHR$（コード）

反対に，文字を ASCII コードに変換するには次のように書く．

ASC（文字列型定数または変数名）

文字列の長さが2字以上の場合，先頭の1字のみが変換される．

（例）　A の ASCII コードは65であるから，

PRINT CHR$(65)

を実行すれば文字Aが出力され

PRINT ASC("A")

を実行すれば数値65が出力される．

［簡単な使用例］

```
10 INPUT C$
20 A=ASC(C$)
30 PRINT "ﾓｼﾞ ";C$;" ﾉ ASCII ｺｰﾄﾞ ﾊ ";A;" ﾃﾞｽ｡"
40 PRINT : PRINT
50 GOTO 10
```

```
RUN
? A
ﾓｼﾞ A ﾉ ASCII ｺｰﾄﾞ ﾊ  65  ﾃﾞｽ｡


? a
ﾓｼﾞ a ﾉ ASCII ｺｰﾄﾞ ﾊ  97  ﾃﾞｽ｡


? %
ﾓｼﾞ % ﾉ ASCII ｺｰﾄﾞ ﾊ  37  ﾃﾞｽ｡
```

5.7 文字列↔数値の変換

文字列型変数は，本来，文章や単語を扱うためのもので，文字列としては英字またはカナ文字が並んでいるのが普通である．しかし，場合によっては，文章の一部に数字が並んでいて，それを数値変数に変換して計算に使いたいとか，逆に，数値変数の値を10進法表現の数字列に変換して文字列の一部分に組み込みたい，ということがある．これを簡単に実行できるように次の二つの関数が用意されている．

文字列型で表されている数値を数値型に変換するには

数値型変数名＝**VAL**（文字列型変数名）

数値型で表されている数値を文字列型に変換するには

文字列型変数名＝**STR$**（数値型変換名）

と書けばよい．

（例）

```
10 A$="ﾜﾀｸｼﾊ 12ｻｲ ﾃﾞｽ"
20 Y$=MID$(A$,7,2)
30 NENREI=VAL(Y$)
40 RAINEN=NENREI+1
50 NY$=STR$(RAINEN)
60 B$="ﾗｲﾈﾝ ﾊ"+NY$+"ｻｲ ﾆ ﾅﾘﾏｽ"
70 PRINT A$
80 PRINT B$
```

```
RUN
ﾜﾀｸｼﾊ 12ｻｲ ﾃﾞｽ
ﾗｲﾈﾝ ﾊ 13ｻｲ ﾆ ﾅﾘﾏｽ
```

[備考]　1）関数**VAL**の，より実用的な応用例が114ページにある．

2）関数**VAL**は，一般に，正負の符号や小数点や指数部の付いた数値でも変換することができる．

3）関数**STR$**の引数は倍精度実数型でもよく，その場合，正しく倍精度の変換がなされる．

5.8　16　進　法

16進法というのは16箇の数字

0　1　2　3　4　5　6　7　8　9　A　B　C　D　E　F

を用い，16^n を表記単位として数を表すやりかたである．ここで，

Aは10

Bは11

Cは12

Dは13

Eは14

Fは15

を表す．また位どりは

1が$16^0 = 1$

10が$16^1 = 16$

100が$16^2 = 256$

1000が$16^3 = 4096$

…

を表す，たとえば，

A 7 5 F

は10進法の

$10\times16^3+7\times16^2+5\times16^1+15\times16^0 = 42847$

を表す．コンピューターの関係で16進法がよく用いられるのは，

コンピューターの内部では2進法が使われている

しかし2進法で数値を表すと桁数が多くなり人間には不便

そこで2進法表現を4桁ずつまとめて16進法で扱うと便利

という理由による．

BASIC による書き方　「この値は16進法で書いてある」ということを表すのに，BASIC では頭に&H という記号を付けて表す．

（例） **&H100**　これは16進法の100だから10進法の256のこと．

&HABC　これは文字の ABC ではなくて16進法の

$$A \times 16^2 + B \times 16^1 + C$$

すなわち

$$10 \times 256 + 11 \times 16 + 12 = 2748$$

のこと．

内部表現（2 進法の数）を 16 進法表現の文字列に直す方法　これには関数**HEX$**を用いればよい．　書き方は

HEX$（数値変数名または式）

（例） 16進法の 1 桁乗算表（10進法の九九の表に相当するもの）を作成するプログラムの例とその出力例を以下に示す（行50の**USING**については6.6節で説明する）．

```
10 FOR I=1 TO &HF
20    FOR J=1 TO &HF
30       IJ=I*J
40       H$=HEX$(IJ)
50       LPRINT USING"&  &";H$;
60    NEXT J
70    LPRINT
80 NEXT I
```

```
1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
2   4   6   8   A   C   E   10  12  14  16  18  1A  1C  1E
3   6   9   C   F   12  15  18  1B  1E  21  24  27  2A  2D
4   8   C   10  14  18  1C  20  24  28  2C  30  34  38  3C
5   A   F   14  19  1E  23  28  2D  32  37  3C  41  46  4B
6   C   12  18  1E  24  2A  30  36  3C  42  48  4E  54  5A
7   E   15  1C  23  2A  31  38  3F  46  4D  54  5B  62  69
8   10  18  20  28  30  38  40  48  50  58  60  68  70  78
9   12  1B  24  2D  36  3F  48  51  5A  63  6C  75  7E  87
A   14  1E  28  32  3C  46  50  5A  64  6E  78  82  8C  96
B   16  21  2C  37  42  4D  58  63  6E  79  84  8F  9A  A5
C   18  24  30  3C  48  54  60  6C  78  84  90  9C  A8  B4
D   1A  27  34  41  4E  5B  68  75  82  8F  9C  A9  B6  C3
E   1C  2A  38  46  54  62  70  7E  8C  9A  A8  B6  C4  D2
F   1E  2D  3C  4B  5A  69  78  87  96  A5  B4  C3  D2  E1
```

5.9 日付と時刻

X1の本体には時計が内蔵されていて

今日の日付は　**DATE$**

現在時刻は　**TIME$**

曜日は　**DAY$**

という文字列型変数の値として読み出すことができる．

本機の時計は主電源を切るとクリアされてしまうので，主電源を入れた時には，その日の日付と現在時刻と曜日を

DATE$="年年／月月／日日"

TIME$="時時：分分：秒秒"

DAY$="曜日"

（必ず2桁の形で入れること）

の形でセットしてやる必要がある．ただし曜日は**SUN**，**MŌN**，**TUE**，**WED**，**THU**，**FRI**，**SAT**で表す．主電源を切らなければ，翌日には日付が変わり，12月31日の次には翌年の1月1日になる．

計算結果やプログラムをプリンターに出力するとき，ついでに日付をプリントしておくと，後日，整理に役立つことが多い．

日付や時刻が文字列型変数になっているのは，主に表示や印字に用いられるからであるが，5.7節で説明した関数**VAL**を用いれば，年，月，日，時，分，秒をそれぞれ別の数値として取り出すことができる．

（例）

```
10 NEN=VAL(MID$(DATE$,1,2))
20 GATU=VAL(MID$(DATE$,4,2))
30 HI=VAL(MID$(DATE$,7,2))
40 ZI=VAL(MID$(TIME$,1,2))
50 HUN=VAL(MID$(TIME$,4,2))
60 BYOU=VAL(MID$(TIME$,7,2))
```

なお，ストップ・ウォッチのように，経過時間の計測に使いたい場合は，**TIME$**とは独立に**TIME**という変数があり，計時開始時点でここに0を入れておいて（代入文**TIME=0**による）終了時点に読み出せば（たとえば**PRINT TIME**）とすれば経過時間（単位は秒）がわかる．

6 入出力文法詳説

6.1 はじめに

入出力に関しては，既に3章でINPUT文とPRINT文について説明した．普通の用途には，この二つを覚えていれば十分で，たいていのことはそれだけで用が足りる．

しかし，少し高度な使い方をしようとすると「こういうことができればよいが…」とか「もっと簡単に書ければ便利なのに…」ということが出てくる．その場合，他のプログラミング言語だと，たいてい「そういうことは文法上できない」「甘ったれたことを言うな」と門前払いを食わせるところであるが，パソコンのBASICは，その点，行きとどいていて，まあたいていのことならばできる仕掛けになっていて，機械語やアセンブラーを使わなくてもパソコンの機能をフルに活用できるようになっている．

それだけに文法はちょっと複雑であるが，最初から全部覚えなくてもよく，だいたいどんなことができるかをざっと目を通しておいて，あとは必要になったときに必要な事項だけ勉強すればよい．

この章では，

　　入　力

　　出　力

　　ファイル入出力

の順に説明する．

6.2　READ 文と DATA 文

これまでのプログラム例では，データーの入力方法として，

　　入力の指示をINPUT文で書く

　　実際のデーターは実行時点に入れる

という方法をとってきたが，もう一つの方法として

　　データーをプログラムと一緒に入れておく

　　その入力指示をREAD文で書く

ということができる．すなわち，入力すべきデーターを

　　DATA　1番目のデーター，2番目のデーター，…

　　（これで書ききれなければ，同じ形式で何行でも並べる）

の形でプログラムの中（普通はプログラムの最後の所）に書き，それをREAD文で読む．READ文の書き方はINPUT文と同じ形

　　READ　変数名，変数名，…

である．INPUT文とREAD文の違いは

　　INPUT文は，実行時に端末機からデーターを読む

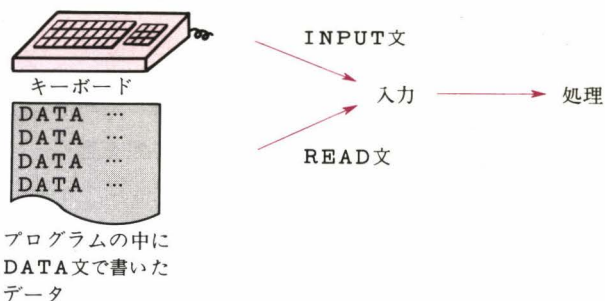
　　READ文は，DATA文で書いたデーターを読む

という点にある．DATA文とREAD文の組合せによる方法には次のような利点がある．

1）統計の計算などの場合，同じデーターをいろいろな手法で解析したいことがある．そういうとき同じデーターを何度も入力しないで済む．

2）デバックのとき，テスト・データーを何度も入れるのはめんどうであるが，この方式なら一度入れるだけでよい．

3）この方式だと，RESTŌRE文を用いて「いまのデーターの最初の位置にもどれ」という指示ができる．こうして，一つのプログラムの中で，同じデーターを何回も読むことができる．

要するに，INPUT文で入力したデーターはすぐ消えてしまってあとに残らないのに対し，DATA文で書いたデーターはプログラムの一部として記憶装置に残っているので，同じデーターを何度も利用することができるわけである．

READ 文の書き方

READ　変数名，変数名，…

DATA 文の書き方

DATA　値，値，…

DATA文はプログラムの中のどこに書いてもよい．

（例1）
```
10 READ A,B
20 C=A+B
30 PRINT C
40 DATA 2.87,-63.1
```

（例2）
```
10 READ A,B
20 C=A+B
30 PRINT "C=";C
40 READ D,E,F
50 G=D*E/F
60 PRINT "G=";G
70 DATA 2.87,-63.1
80 DATA 0.56
90 DATA 0.02
100 DATA -8.1
RUN
C=-60.23
G=-1.38272E-03
```

（例3）　プログラムの途中にDATA文を書いてもよい．

```
10 READ A,B,C
20 DATA 1,-5,6
30 X=(-B+SQR(B^2-4*A*C))/(2*A)
40 PRINT X
RUN
 3
```

6.3　INPUT$

これは，キーボードから指定字数だけ読み込むための機能で，文法上は関数の扱いになっており，その引数の所に指定字数を書く．

標準的な使い方は

文字列型変数名＝INPUT$（字数）

（例）　A$＝INPUT$（6）

INPUT文との主な違いは

1）命令ではなくて関数である．

2）字数指定方式である．

3）入力文字列の終りに [↵] キーを押す必要がない．

4）入力要求を示す？印もカーソルも表示されない．

5）入力した文字が画面に表示されない．

6）文字以外のキー，たとえば

[↵]　[←]　[↑]　[↓]　[→]　[ESC]　[INS]　[DEL]

なども，ASCIIコードの形で読み込むことができる．

7）誤ってキーを押した場合，カーソルをもどして訂正することができない．

制御キーが内部でどんなコードになるかを調べるには

```
10 A$=INPUT$(1)
20 K=ASC(A$)
30 PRINT K
```

というようなプログラムを走らせて，いろいろなキーを押してみればよい．ファンクション・キー [F1] ～ [F5] を押すとキーに登録されている文字列（AUTOなど）の先頭から指定字数が読み込まれる．

例題 1 ―― タイプ練習

A, B, C, …のキーを順に押すように指示し，押されたキーを調べ，まちがっていたらエラー・メッセージを出して再入力させるプログラムを作れ.

[**解答**]

```
10 PRINT "ABC...ヲ イレテゴ ラン"
20 TIME$="00:00:00"
30 FOR K=65 TO 90
40     A$=CHR$(K)
50     B$=INPUT$(1)
60     PRINT B$;
70     IF A$=B$ GOTO 100
80     PRINT:PRINT "チガ ウ!"
90     GOTO 50
100 NEXT K
110 PRINT:PRINT "タイム";TIME$
```

```
RUN
ABC...ヲ イレテゴ ラン
ABCDEFGHIJKLN
チガ ウ!
MNOPQRT
チガ ウ!
STUVWXYZ
タイム00:00:22
```

[**備考**]　1）上のプログラムは大文字として入力することを前提にしているから，CAP キーを押して大文字シフトにロックしてから練習すること.

2）**INPUT$** は関数であるから，**IF** 文の条件式の中や，代入文の文字列式の中に書くことができる．上のプログラムも，エコーを表示する必要がなければ，次のように短くなる.

```
10 PRINT "ABC...ヲ イレテゴ ラン"
20 TIME$="00:00:00"
30 FOR K=65 TO 90
40     IF INPUT$(1)<>CHR$(K) THEN BEEP:GOTO 40
50 NEXT K
60 PRINT "タイム";TIME$
```

6.4　INKEY$

これは「キーボードの現在の状態*」を調べる関数で,

キーを押していれば，その文字が入力される

キーを押してなければ，空列（""）が入力される

関数であるが引数は不要で，普通,

文字列型変数名=INKEY$

(例)　A$=INKEY$

の形で用いる.

前節で説明したINPUT$と同様

⏎キーを押す必要がない

?印もカーソルも表示されない

入力した文字が画面に表示されない

文字以外のキーを入力することができる

などの点がINPUT文と異なり，そういう意味では

INPUT$(1)

とよく似ているが，INPUT$はキーを押すまで待つのに対し，INKEY$は，キーを押してなければ「押してない」という情報を返して先に進む，という点が異なる.

---

*　主旨としては，本文の説明どおり「現在の状態」を調べたいのであるが，じつは入力バッファーという待合室のような所があって，キーボードから入ってきた文字は，そこに並んで入力命令を待つ，という方式になっている．普通は入力として要求されたデーターだけを入力するから，キーボードから入ってきた文字は直ちに処理され，入力バッファーは，たいてい，空になっている．しかし誤操作やイタズラでキーボードから余分な文字が入っていると，まず入力バッファーの内容が読み込まれるため，必ずしも「現在の状態」に一致しないことがありうる.

6.5 LINE INPUT

これは[⏎]キーが押されるまでを一つの文字列として入力する命令で，次の形式で書く．

LINE INPUT 文字列型変数名

または

LINE INPUT "入力要求", 文字列型変数名

普通の**INPUT**文との違いは

- ，や：や"印を含む文字列をそのまま入力できる
- 読み込み先の変数名は一つしか書けない

(例) **LINE INPUT A$,B$** は誤り．

- 入力待ちを示す？印は表示されない（"入力要求メッセージ"のあとを；印にしても？印は出ない）．

などである．

典型的な用途としては，文章の入力，記号列（数式など）の入力，プログラムの入力（たとえばプリ・プロセッサーをBASICで書く場合）などがある．

オート・リピート機能のON, OFF　キーボード入力のとき，キーを長く（約1/2秒以上）押していると，同じキーを何回も続けて押したのと同じ効果になる（まだ知らなかった人は実際にすぐやってみるとよい．たとえば**A**のキーを長く押していると，画面上に**AAAAAA**…と表示される）．これをオート・リピート機能といい，カーソルの移動その他に便利なものであるが，**INPUT$**や**INKEY$**の際にはジャマになることがある．そういう場合は[CTRL]キーと[SHIFT]キーを押しながら[0]のキーを押せばよい．オート・リピートを再開したいときは[CTRL]と[SHIFT]と[1]のキーを押す．

6.6 PRINT USING

これは出力書式を詳しく指定したい場合に用いる文で

PRINT USING"書式指定";出力項目の列

の形で書く．書式は次のような形で指定する．

+	符号（実際には数値の頭に付く）	.	小数点を出力する位置
#	数字を出力する位置	空白	空白にしておく位置

（例）　PRINT USING"+##.### ";A;B

これは，「AとBを

符号は数値の前に付ける．正は+，負は−で表わす

整数部に2桁とる

左から4字目に小数点を打つ

小数部に3桁とる

その右に1字分の空白を入れる

という形で出力せよ」ということを表す．

1）先頭の位置に「#」を書くと，正の場合の符号表示が+印でなく空白になる．

2）符号を右に付けることもできる．それは+か−を右に書くことによって指定する．「−」印は上記1）と同様，「+サプレス」の意味．

（例）　PRINT USING"###.##+";X

3）整数値として右づめで表示したい場合は「.」印は不要．

（例）　PRINT USING"+###";N

4）3桁ごとにコンマを入れることも可能．

（例）　PRINT USING"+#,###,###";KINGAKU

5）指数部付きの形（浮動小数点表示）で出力したい場合は，右に^^^^を付ける．指数部はE±××の形で出力されるので常に4桁必要である．

6）文字列を出力する場合は，次のように書く．

（例）　PRINT USING"&␣␣␣&",A$

この場合&も含めた字数分の出力場所が確保され左づめで出力される．

[プログラム例と出力例]

```
10 A=234.56
20 PRINT USING"+###.##";A        +234.56
30 PRINT USING"####.##";A         234.56
40 PRINT USING"+####.#";A         +234.6
50 PRINT USING"#####.#";A          234.6
60 PRINT USING"+######";A           +235
70 PRINT USING"#######";A            235
80 PRINT USING"+#.####";A        %+234.5600
```

最後の%印は桁数オーバーのため指定書式どうりに出力できないことを表す.

```
10 A=.00123
20 PRINT USING"#.##^^^^";A
RUN
0.12E-02
```

7) 先頭に¥¥と書くと数値の頭に¥印が付く.

8) 先頭に**と書くと数値の左側の欄内余白は*印になる.

9) 先頭に*¥と書くと,数値の頭に¥印が出力され,欄の左端から¥印までの余白は*印でうめられる.

10) &空白 n 箇&　と書くと, $n+2$ 字(いいかえれば両端の&も含めて数えた字数)分の表示場所が確保され,文字列変数(一般に文字列式)の値(文字列)が左づめで出力される(逆に,文字列の方が長い場合は,先頭の n 字が出力される).

11) 書式制御用以外の文字はそのまま出力される.

[プログラム例と実行例]

```
10 A$="ABCDEF"                         RUN
20 B$="PQR"                            ABC
30 PRINT USING"& &";A$                 PQR   ABCDE
40 PRINT USING"&    &";B$;A$
```

6.7　TAB と SPC

これらはPRINT（またはLPRINT）文の中で用いられ，

TAB（*n*）は「左から *n* 字目の位置までスキップせよ」

SPC（*n*）は「*n* 字分の空白を出力せよ」

ということを表す．

（例）

```
10 PRINT "01234567890123456789O"
20 A=-753
30 PRINT A;TAB(12);A
40 PRINT TAB(3);A;TAB(12);A
50 PRINT TAB(5);A;TAB(12);A
```

```
RUN
01234567890123456789O
-753          -753
   -753       -753
     -753     -753
```

［注意］ TABによって現在位置より左にもどることはできない．あえてそのようなプログラムを書けば，次の行の指定位置に次の項目を出力する．

（例）

```
10 PRINT "012345678901234567890123456789O"
20 A=-753
30 PRINT TAB(12);A;TAB(3);A
```

```
RUN
012345678901234567890123456789O
            -753
   -753
```

6.8 STRING$

これは文字列型の関数で，普通は

STRING$(字数,"文字")

の形で書き，第2引数に書かれた文字を，第1引数で指定された箇数だけ並べたものを表す．

(例) **STRING$(3,"K")** は **"KKK"** と同等

一般には

STRING$(数値型変数名,文字列型変数名)

でもよい．第2引数で指定された文字列が2字以上の場合，先頭の文字のみが有効である．

(使用例) **STRING$** は，簡単に棒グラフを描かせたい場合に便利である．以下に示すのは sin x のグラフを **STRING$** で描かせた例である．

```
10 FOR X=0 TO π STEP π/20
20     LPRINT STRING$(30*SIN(X),"*")
30 NEXT X
```

```
*****
*********
**************
******************
*********************
************************
***************************
*****************************
******************************
******************************
******************************
*****************************
***************************
************************
*********************
******************
**************
*********
*****
```

6.9 カセット・テープへの記録と再生

カセット・テープにデーターを記録するには，最初に

```
ŌPEN "Ō",#番号,"ファイル名"
```

によって，カセット・テープを出力に使う旨を宣言し，

```
PRINT #番号,出力項目の列
```

によって出力し，最後に

```
CLŌSE #1
```

によって，データーの終りを表す印をテープに書き込む．こうして記録したテープを再生するには，まず

```
ŌPEN "I",#番号,"ファイル名"
```

によって，カセット・テープを入力に使う旨を宣言し，

```
INPUT #番号,入力項目の列
```

によって入力し，最後に

```
CLŌSE #1
```

によって，終了した旨をコンピューターに知らせる．

[備考]　1）ファイル名は13字以内で大文字，小文字，数字，カナ文字，記号（コンマ，コロン，セミコロン以外）を使用できる．

2）番号は1から15まで使えるが，**MAXFILES**文(205ページ参照)で指定した値（指定しなければ1）を越えてはいけない．

例題 2 ―― データーの記録，再生

データー $a_1, a_2, \cdots, a_n$ をキーボードから読み込み，カセット・テープに記録するプログラム，およびそれを再生するプログラムを作れ．

[解答] データーをキーボードから入力するには，かなりの手間がかかるから，一度入れたデーターは，なるべくカセット・テープかディスクに記録して保存しておく方がよい．ここに示すのは，そのようなプログラムのひな型である．後日，データーを識別するために，何かタイトル(表題)を付けておくことにする．データーの箇数**N**も記録しておく方が便利である．ファイル名はキーボードから入力するようにした．こうしないと同じ名前のファイルがたくさんできてしまう．

(記録)

```
10 REM ------- input -------
20 INPUT "ﾀｲﾄﾙﾊ?",TITLE$
30 INPUT "ﾃﾞｰﾀｰ ﾉ ｺｽｳﾊ?",N
40 DIM A(N)
50 FOR I=1 TO N
60     INPUT A(I)
70 NEXT I
80 REM ------- save -------
90 INPUT "ﾌｧｲﾙ ﾒｲ ﾊ?",F$
100 OPEN "O",#1,F$
110 PRINT #1,TITLE$
120 PRINT #1,N
130 FOR I=1 TO N
140     PRINT #1,A(I)
150 NEXT I
160 CLOSE #1
170 END
```

(再生)

```
10 REM ------- load -------
20 INPUT "ﾌｧｲﾙ ﾒｲ ﾊ?",F$
30 OPEN "I",#1,F$
40 INPUT #1,TITLE$
50 INPUT #1,N
60 FOR I=1 TO N
70     INPUT #1,A(I)
80 NEXT I
90 CLOSE #1
```

6.10 文字の特殊な表示方法

点滅させる　文字を点滅させたいときは

CFLASH 1

と命令する．この命令実行後に表示される文字は点滅表示となる．普通の表示にもどすときは

CFLASH 0　　（または単に**CFLASH**）

とする．

反転させる　文字を

A B C D E F G

1 2 3 4 5 6 7

のように反転表示したいときは

CREV 1

と命令する．普通の表示にもどすには

CREV 0　　（または単に**CREV**）

とすればよい．

拡大する　文字を2倍に拡大して表示することができる．

CSIZE 0　　普通の大きさ

CSIZE 1　　縦方向のみ2倍に拡大

CSIZE 2　　横方向のみ2倍に拡大

CSIZE 3　　縦，横とも2倍に拡大

CSIZEを用いる場合は　PRINT #0　という文で出力しなければいけない（#0のŌPEN，CLŌSEは不要）．

色を付ける　**CŌLŌR**文を用いる．7.5節参照．

画面上の指定位置に表示　**LŌCATE**によってカーソルを指定位置に移し，それから**PRINT**文で出力すればよい．位置の指定は

LŌCATE 左から何字目，上から何行目

と書く．ただし，横位置，縦位置は左端，上端が0である．

7 図形表示

7.1 機能紹介

図形を表示するには，最低限，次のことができればよい．

　　画面を消す．

　　指定された位置に点（ドット）を表示する．

これさえできれば，あとは工夫しだいでどんな図形でも描けるが，

　　指定された2点を直線で結ぶ

　　指定された点を中心とし，指定された半径で円を描く

　　指定された位置に，指定された文字を書く

などの機能があれば，さらに便利であろう．また（カラー・グラフィック・ディスプレイやプリンターをもっていれば）

　　色の指定

　　プリンターへの出力

などもできるようにする必要がある．

本機には，このために次のような命令がある．

PSET	点を表示
PRESET	点を消す
LINE	直線を引く
CIRCLE	円を描く
CŌLŌR	色の指定
PAINT	色をぬる
HCŌPY	プリンターに出力する

そのほか，スケーリング（寸法調節）やズーミング（定義した図形の一部分のみを拡大して表示）などの機能もある．

7.2　オープニング・セレモニー

本機の場合，LINE文やCIRCLE文をいきなり使ったら，何も表示されない．グラフィック関係の命令を使うには，SCREEN文によって「グラフィック・モードの宣言」をしておく必要がある．それには，普通，

```
SCREEN 0,0,0
```

と書けばよい*．パラメーターとして0，0，0以外のものを書くこともできるが，それは特殊な用法である（7.13節参照）．もっと簡単に

```
INIT
```

と書いてもよい．INITというのは本機独特の命令で「画面表示関係の指定をすべて標準状態にもどせ」という意味をもつ**．前に誰かが（または自分が）グラフィック関係のプログラムを実行した場合には，なるべくINIT文によって標準状態にもどしてから使う方が安全である．

---

*　GRAPH　0,0,0　と書いてもよい．GRAPHとSCREENは同一の命令である．

**　INITはinitialize（初期化）の意味である．3.16節で説明したように，RUNコマンドで実行を開始すれば変数の名はすべて初期化される（数値変数の値は0になり，文字列型変数の中味は空になる）が，画面表示関係の指定は継続するので，初期化したければINIT文によって行なう必要がある．なお，INIT文で初期化されるのは，CŌLŌR，CGEN，CFLASH，CREV，CSIZE，PALET，PRW，WINDŌW，CŌNSŌLE，SCREENの各命令の指定をもとにもどすだけであって，WIDTH文の指定は変更されず，また画面消去もされない．

7.3 画 面 消 去

画面消去はCLS文によって行なう.

CLS	文字だけを消す
CLS 0	グラフィックだけを消す
CLS 4	グラフィックと文字を同時に消す

画面の一部分しか消えない場合　後述のWINDŌW文, CŌNSŌLE文などによって, 表示領域が制限されていると一部分しか消えない.

```
WINDŌW (0,0)-(319,199)
CŌNSŌLE 0,25
```

(パラメーターを省略して,

```
WINDŌW:CŌNSŌLE
```

としてもよいようである. 前節で説明したINIT文を用いてもよい)によって領域制限を解除してからCLS文を実行すればよい.

```
(正) WINDŌW:CŌNSŌLE
     CLS 4
(誤) CLS 4
     WINDŌW:CŌNSŌLE
```

全然消えない場合　後述のマルチ・ページ機能を用いている場合, CLS文は「現在書き込み中のページ」だけしか消去しない.

```
SCREEN 0,0
CLS 4          } ページ0を消去
SCREEN 1,1
CLS 4          } ページ1を消去
```

とすれば全部消える. ただし, 上記のままだとページ1の指定になってしまうから, ページ0を使用したければ, あらためて

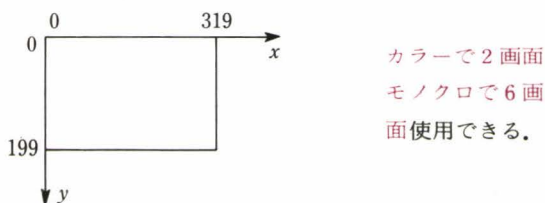
```
SCREEN 0,0
```

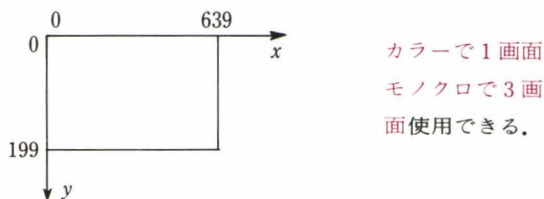
とする必要がある.

7.4 座　標　系

本機では次の2種類の画面座標系を用いることができる．

1）WIDTH 40　と指定すると，1行の文字数は40字，図形表示は320×200ドットとなり，画面座標系は次のようになる．

カラーで2画面
モノクロで6画面使用できる．

2）WIDTH 80　と指定すると，1行の文字数は80字，図形表示は640×200ドットとなり，画面座標系は次のようになる．

カラーで1画面
モノクロで3画面使用できる．

［解説］　1）と2）のどちらを使おうかと迷うかもしれないが，筆者は1）の方を推奨する．1）だと縮尺比（画面座標1単位の実際に表示される長さ）が縦方向，横方向ともだいたい同じぐらいなので，320×200 mmの方眼紙（1 mm目盛）のつもりで画面を使えば，だいたい思いどおりの形を描くことができる．それに対し，2）だと縦横の縮尺比が違うので，いろいろとやっかいである．

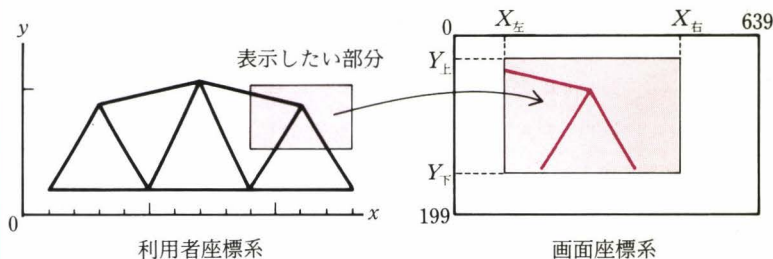
「1）の方は表示が粗いかも……」と心配される向きがあるかもしれないが，320×200ドットといえばMZ-80Bと同じで，実用上はこれで十分である．横方向を640ドットにするなら縦方向も倍の400ドットにしなければ意味がなく，そうすれば確かに優雅な曲線が描けるが，640×400ドットの表示には，もう1ランク上のパソコンと高価なディスプレイ装置が必要である．

利用者座標系　応用の際には，利用者は個々の目的に適した座標系を用いるのが普通である．これを利用者座標系と呼ぶことにしよう．その値は負数になるかも知れないし，非常に大きな値（たとえば10^8）になるかもしれない．他のパソコンでは，たいてい，画面座標しか使えないので，利用者座標系から画面座標系への変換は利用者が行なう（プログラムを書く）必要があったが，本機では図形表示の命令に利用者座標の値をそのまま用いることができる（座標変換はコンピューターがやってくれる）．

WINDOW 文　利用者座標と画面座標の対応関係は，**WINDŌW** 文で指定する．基本的な書き方は

WINDŌW $(X_{左}, Y_{上})-(X_{右}, Y_{下}), (x_{左}, y_{上})-(x_{右}, y_{下})$

画面座標系　利用者座標系

で，このように書けば利用者座標系の x の変域$[x_{左}, x_{右}]$を画面座標系の$[X_{左}, X_{右}]$に，また y の変域$[y_{下}, y_{上}]$を画面座標の$[Y_{下}, Y_{上}]$に変換して表示してくれるわけである．

[解説]　上の図にあるように，利用者画面上で描いた図形の内の指定された部分だけを表示するテクニックはシザリング(scissoring，はさみで切り抜くこと）といって，このような機能をもたないコンピューターでソフトウェア的に実現しようとするとかなりむずかしい．

本機はこういう機能をもっているので，後述の**LINE**文や**CIRCLE**文で図形の一部が画面の外に出てもエラーにならず，正しく実行される．ただし，極端に画面をはずれるとエラー・メッセージが出たり停止したりすることがある．

7.5 色の指定

色番号　色は基本的には次の色番号で指定する．

0	1	2	3	4	5	6	7
黒	青	赤	紫	緑	水色	黄	白

COLOR 文　文字の色と背景の色はCŌLŌR文によって指定する．図形の色は，図形表示命令（LINE文，CIRCLE文等）で指定できるが，「指定を省略したときの色」はCŌLŌR文の指定に従う．

> CŌLŌR文の書き方は
>
> 　　CŌLŌR　文字や図形の色
>
> または
>
> 　　CŌLŌR　文字や図形の色，背景の色

［備考］ 1） CŌLŌR文は「これから表示する文字や図形」に対してのみ有効である．既に表示されている図形や文字は変化しない．

2） 背景の色はCŌLŌR文の実行と同時に変る．

3） 文字の色と背景の色は，上に示した「基本的な色番号」でしか指定できない（後述のPALET番号は図形に対してのみ有効）．

4） 電源投入時には　CŌLŌR 7，0　の状態になっている．INIT文を実行すれば，この状態にもどる．

5） 文字の色と背景の色の組合わせできれいなのは

　　CŌLŌR 7，1　　青地に白
　　CŌLŌR 7，2　　赤地に白
　　CŌLŌR 1，5　　水色地に青

などである．CŌLŌR 0，7　とすると「白地に黒」になりそうであるが，実際には何も見えなくなってしまう．

6） 背景の色は手動操作で変えることもできる．それには CTRL キーを押しながら色番号の数字キーを押せばよい．

パレット番号による色の指定　左記の色番号は他の多くのパソコンでも用いられている標準的なコードであり，特に指定をしなければ色番号と表示される色の対応は左記のようになるが，もし必要があれば，この対応関係（色コード表）を変更することができる．このような「ユーザーが決めた色番号」のことをパレット番号という．

> パレット番号と色番号の対応は**PALET**文によって指定する．
>
> 　　PALET　パレット番号，色番号
>
> パレット番号としては0～7を指定できる．

（例1）

```
PALET 1,2
PALET 2,6
PALET 3,4
PALET 4,5
PALET 5,1
PALET 6,3
```

とすれば，虹色の順の色番号（パレット番号）

0	1	2	3	4	5	6	7
（黒）	赤	黄	緑	水	青	紫	（白）

になる．

（例2）

```
PALET 3,0
```

とし，他の色番号は変更せず，背景は0（黒）とすれば，色番号3で描かれた図形を見えなくすることができる．あとで

```
PALET 3,3
```

とすれば，再び見えるようになる．

［備考］　**PALET**文を実行すると，既に表示されている図形の色も全部，瞬時に変る．文字の色は**PALET**文を実行しても変化しない．

7.6 直線を引く

直線を引くには，その両端点の座標を指定して，普通，

LINE （横座標，縦座標）－（横座標，縦座標），PSET

と書く*．第1の（　）内は始点，第2の（　）は終点の座標である．なお，前の線分の終点が次の線分の始点となる場合は始点を省略して

LINE －（横座標，縦座標），PSET

と書いてよい．座標は利用者座標を使用できる．

（例） LINE （20，80）－（100，30），PSET
　　 LINE －（130，50），PSET

線の消去　画面全体を消さずに特定の線だけ消したい場合は

LINE （横座標，縦座標）－（横座標，縦座標），PRESET

と書く．第1の（　）内は始点，第2の（　）内は終点の座標である．

点線や破線の描き方　LINE文のPSETの後に

，色番号，ライン・スタイル

の形でライン・スタイルというのを指定できる．これは，点線，破線などを16ビットのビット・パターンで表したもので，たとえば

—　————————　—(以下これのくりかえし)

という線を引きたいのであれば，これをビットで表すと

1001111111111001　　2進法表現
　9　 F　 F　 9　　16進法表現

であるから，ライン・スタイルは&H9FF9となる（5.8節参照）．

（例） LINE（0，0）－（30，20），PSET，6，&H9FF9

---

* 中央の－印は，意味としてはハイフンであるが，キーボードにはハイフンに当たる字がないので，負号（マイナス）を用いる．

7.7 四角を描く

LINE文にはいろいろなオプションを付けられる．まず，基本形

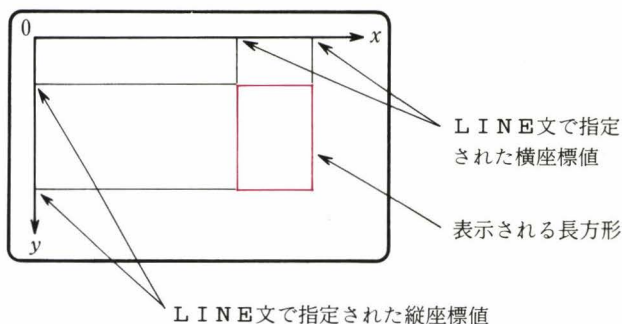
LINE　（横座標，縦座標）―（横座標，縦座標），PSET

の次に

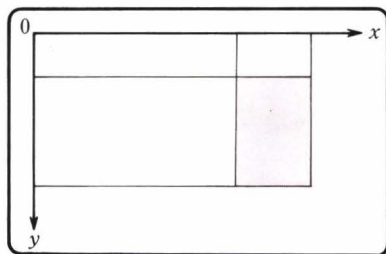
，色番号

を書くことにより，線の色を指定できる（色番号→7.5節参照）．またその次に

，B

を付けると，普通のLINE文のような直線は引かず，かわりに，

のような長方形が表示される．「，B」のかわりに

，BF

とすると，上記の四角の中がぬりつぶされる．

また，PSETのかわりにPRESETと書くと，直線または四角が消去される．

7.8 折れ線を描く

複数個の点を結んで，多角形や折れ線グラフを描くには，**LINE**文を何回も使えば書けるが，本機では次のように簡単に書くことができる．

書き方は

　　LINE　$(x_1, y_1)-(x_2, y_2)-\cdots-(x_n, y_n)$

である．(x, y)は各点の座標である．このあとに

　　，色番号

を付けて色の指定をしたり，さらに

　　，ライン・スタイル

を付けて点線等の指定をすることもできる．

[使用例]　矢印を描くプログラムを作ってみよう．

```
7000 REM ﾃﾝ(X1,Y1)ｶﾗ(X2,Y2)ﾆﾑｶｳ ﾔｼﾞﾙｼﾖｶｸ
7010 LABEL "ARROW"
7020 DX=X2-X1 : DY=Y2-Y1
7030 EL=SQR(DX*DX+DY*DY)
7040 CA=DX/EL : SA=DY/EL
7050 T=π*5/6
7060 CB=COS(T) : SB=SIN(T)
7070 R=5
7080 X3=X2+R*(CA*CB-SA*SB)
7090 Y3=Y2+R*(SA*CB+CA*SB)
7100 X4=X2+R*(CA*CB+SA*SB)
7110 Y4=Y2+R*(SA*CB-CA*SB)
7120 LINE(X1,Y1)-(X2,Y2)-(X3,Y3)-(X4,Y4)-(X2,Y2)
7130 RETURN
```

[メイン・プログラムの例]

```
10 X1=30 : Y1=20
20 X2=70 : Y2=100
30 SCREEN 0,0,0
40 CLS 4
50 GOSUB "ARROW"
60 END
```

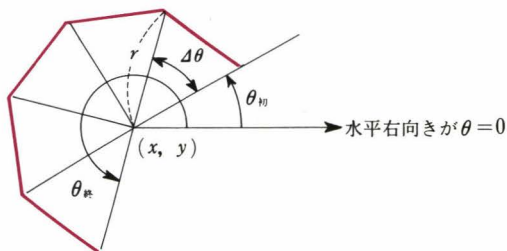
7.9　正多角形や星形を描く

正多角形は**LINE**文を用いて描けるが，本機では**PŌLY**という命令を用いて非常に簡単に描くことができる．指定方法は次のとおり．

PŌLY　(x, y), r, $\varDelta\theta$, $\theta_{初}$, $\theta_{終}$

ただし

x	中心の x 座標
y	中心の y 座標
r	中心から頂点までの距離（外接円の半径）
$\varDelta\theta$	角度増分
$\theta_{初}$	開始角度
$\theta_{終}$	終了角度

（$\varDelta\theta$，$\theta_{初}$，$\theta_{終}$）下図参照．角度の単位は度（degree）

［備考］　1）正 n 角形を描くには

$$\varDelta\theta = 360/n \qquad \theta_{終} = \theta_{初} + 360$$

とすればよい（$\theta_{初} = 90$ とすれば頂点が真上に来る）．

2）いささか変則的な使用法であるが

$$\varDelta\theta = 144 \qquad \theta_{終} = \theta_{初} + 720$$

とすれば星形を描くことができる（$\theta_{初} = 90$ とすれば頂点が真上に来る）．

7.10 円を描く

円を描くにはCIRCLE文を用いる．基本的な書き方は

CIRCLE　(中心の横座標，同縦座標)，半径

で，オプションとして

色番号　　一般にはパレット番号で0〜7の値

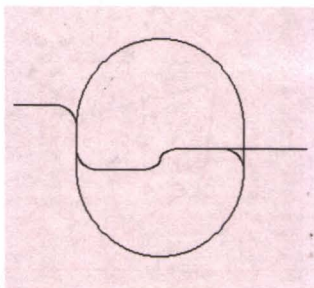
縦横比　　楕円を描くためで（縦径）／（横径）の値*.

開始角度 }　円孤を描くためのもので，右向きを0度とし，反
終了角度 }　時計まわりの角度(度，degree 単位)で指定する．

を指定できる（上記の順にコンマで区切って書く）．

座標は利用者座標系で指定することができる．

［使用例］ 東京の国電の地図を描いてみよう．(挿絵参照)

```
10 REM --- トウキョウ デ゛ンカン ---
20 INIT
30 CLS 4
40 REM ヤマノテセン
50 CIRCLE (160,80),80,4,1,0,180
60 CIRCLE (160,119),80,4,1,180,360
70 LINE (80,80)-(80,119),PSET,4
80 LINE (240,80)-(240,119),PSET,4
90 REM チュウオウセン
100 CIRCLE (100,100),20,2,1,180,270
110 LINE (100,120)-(140,120),PSET,2
120 LINE (180,100)-(220,100),PSET,2
130 CIRCLE (140,110),20,2,.5,270,360
140 CIRCLE (180,110),20,2,.5,90,180
150 CIRCLE (220,120),20,2,1,0,90
160 CIRCLE (60,80),20,2,1,0,90
170 LINE (20,60)-(60,60),PSET,2
180 REM ソウブ゛セン
190 LINE (220,100)-(300,100),PSET,6
200 END
```

＊　ただし640×200ドット（WIDTH　80）の場合は，本当の縦横比の1/2をここに書く（縦横のドット密度が異なるため）．

7.11 点 の 表 示

画面上の指定位置に点を表示するには，次のように書く．

PSET （横座標，縦座標）

座標は利用者座標系を使用できる（ただしWINDŌW，VIEWの指定がなければ画面座標系の値として解釈される．これは後述のLINE文でも同様である）．

色を指定したければ，次のように書くことができる．

PSET （横座標，縦座標），色番号

パレット使用の場合は，この色番号はパレット番号と解釈される．

点の消去　画面全体を消さずに特定の点だけを消したい場合は

PRESET （横座標，縦座標）

を用いる．

[簡単な使用例]

```
10 REM ※※※ ｷﾗｷﾗﾎﾞｼ ※※※
20 INIT
30 CLS 4
40 DEFINT A-Z
50 N=100 : M=10
60 DIM X(N),Y(N),C(N)
70 FOR I=0 TO N
80     X(I)=RND*320
90     Y(I)=RND*200
100     C(I)=RND*6+1
105     PSET(X(I),Y(I),C(I))
110 NEXT I
120 REPEAT
130     I=M*RND
140     PSET(X(I),Y(I),C(I))
150     J=M*RND
160     PRESET(X(J),Y(J),C(J))
170 UNTIL RND=π/5
```

7.12 ぬりつぶす

閉曲線で囲まれた領域を色でぬりつぶすには，領域内の1点（領域内であればどこでもよい*）の座標x,yと，ぬる色の番号（パレット番号による指定可）c，および，境界の色番号bを指定して

PAINT $(x, y), c, b$

と書く．

（例）　赤い三角形を描くには，赤い線で三角形を描き，内部の1点を指定して赤でPAINTすればよい．

```
10 INIT : COLOR 2
20 LINE (50,170)-(120,30)-(260,150)-(50,170)
30 PAINT (160,100),2,2
```

タイリング　本機では，タイリング（tiling，タイル張り）といって，指定した色模様でぬりつぶすことができる．これを行なうにはPAINT文の色番号cのかわりに，青，赤，緑のドット・パターン（各8ビット）を文字列型の定数または変数の形で与える．ドット・パターンは3バイトが一組で，それが横方向のドット・パターンを表し，二組以上指定した場合には，最初の3バイトが1段目，次の3バイトが2段目，…のドット・パターンとなる（n組指定すればn段ごとくりかえす）．（挿絵参照）

（例）　赤の点と紫の点を交互に並べるには，ドット・パターンを

原色	奇数段目	16進表現	偶数段目	16進表現
青	10101010	AA	01010101	55
赤	11111111	FF	11111111	FF
緑	00000000	00	00000000	00

とすればよいから，色番号のかわりに次のように書けばよい．

HEXCHR$("AAFF0055FF00")

---

* ただし始点が最初から色番号cになっていると，ぬりつぶしは行なわれない．

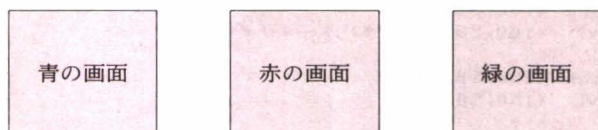
同様にして各種の中間色を作り，並べて表示するプログラムの一例を以下に示す．

```
10 REM --- チュウカンショク ---
20 INIT
30 CLS 4
40 O$=CHR$(&H0)
50 A$=CHR$(&HAA)
60 F$=CHR$(&HFF)
65 G$=CHR$(&H55)
100 REM アカ
110 CIRCLE (50,50),20,7
120 PAINT  (50,50),2,7
130 REM アカムラサキ
140 CIRCLE (100,50),20,7
150 PAINT  (100,50),A$+F$+O$+G$+F$+O$,7
160 REM ムラサキ
170 CIRCLE (150,50),20,7
180 PAINT  (150,50),3,7
190 REM アオムラサキ
200 CIRCLE (200,50),20,7
210 PAINT  (200,50),F$+A$+O$+F$+G$+O$,7
220 REM アオ
230 CIRCLE (250,50),20,7
240 PAINT  (250,50),1,7
300 REM アカ
310 CIRCLE (50,100),20,7
320 PAINT  (50,100),2,7
330 REM ダイダイ
340 CIRCLE (100,100),20,7
350 PAINT  (100,100),O$+F$+A$+O$+F$+G$,7
360 REM キイロ
370 CIRCLE (150,100),20,7
380 PAINT  (150,100),6,7
390 REM キミドリ
400 CIRCLE (200,100),20,7
410 PAINT  (200,100),O$+A$+F$+O$+G$+F$,7
420 REM ミドリ
430 CIRCLE (250,100),20,7
440 PAINT  (250,100),4,7
500 REM アオ
510 CIRCLE (50,150),20,7
520 PAINT  (50,150),1,7
530 REM ウスアオ
540 CIRCLE (100,150),20,7
550 PAINT  (100,150),F$+O$+A$+F$+O$+G$,7
560 REM ミズイロ
570 CIRCLE (150,150),20,7
580 PAINT  (150,150),5,7
590 REM ウスミドリ
600 CIRCLE (200,150),20,7
610 PAINT  (200,150),A$+O$+F$+G$+O$+F$,7
620 REM ミドリ
630 CIRCLE (250,150),20,7
640 PAINT  (250,150),4,7
```

7.13 複数画面の使用法

カラー写真を原色版で印刷するときは，もとの写真を赤，青，黄の3原色に分解し，赤の版，青の版，黄の版を作って刷り重ねる．

カラー・テレビも同様で，テレビ・カメラに入った光を，青，赤，緑の3原色に分解し，それぞれの画像信号を電波に乗せて送り，家庭のテレビのブラウン管の上で合成する．

本機でカラー表示する場合も同様で，

というぐあいに3面分のメモリー（各16Kバイト）があって，それを画像信号になおしてディスプレイ装置に送り，ブラウン管の上で合成している．

7色のカラー表示をするには，上記のようにするほかないが，もしもモノクロ（単色）でよい，というのであれば，青の画面，赤の画面，緑の画面を，それぞれ別の（独立した）画像メモリーとして使うことができる．

画像メモリーをたくさん持っていると，いろいろと面白いことができる．たとえば，赤の画面で何か表示しておいて，その間に緑の画面に新しい画像を書き込んでおいて，表示を緑の画面に切り換えると，パッと新しい画像が表示される（複雑な画像をメモリーに書き込むには，かなりの時間がかかるから，上記のようにしない限り，瞬時に新しい画像に切り換えることはできない）．

また，赤の画面には地図を描いておいて，他の画面に付加情報を描いておいて，重ねて表示する，というような使い方もある．

640×200ドットで表示する場合は，そういうわけで

カラー（7色）表示ならば　1画面（第0ページ）

モノクロ（単色）表示ならば　3画面（第0ページの赤，青，緑）

ということになるが，320×200ドットならば，1画面に必要なメモリーが半分で済むから

カラー（7色）で　2画面（第0ページと第1ページ）

モノクロ（単色）で　6画面（各ページの赤，青，緑画面）

の使用が可能である。

第0ページ	（画面番号1） 青の画面 （色番号1）	（画面番号2） 赤の画面 （色番号2）	（画面番号3） 緑の画面 （色番号4）
第1ページ	（画面番号1） 青の画面 （色番号1）	（画面番号2） 赤の画面 （色番号2）	（画面番号3） 緑の画面 （色番号4）

グラフィック機能を用いる際には，これらの内の

どのページの，どの画面に書き込むか

どのページの，どの画面を表示するか

を指定する必要がある．この内，ページの指定は次のように書く．

```
SCREEN 表示するページ，書き込むページ
```

組合わせは次の4通りしかない．

SCREEN 0,0	第0ページを使用	書き込んで，すぐに表示する．
SCREEN 1,1	第1ページを使用	
SCREEN 0,1	0ページを表示し，1ページに書き込む	
SCREEN 1,0	1ページを表示し，0ページに書き込む	

標準状態は　SCREEN 0,0　で，電源投入時，リセット時，INIT文実行時にはこの状態にもどる．

画面消去命令CLSは，「書き込むページ」に対してのみ作用する．

複数画面の使用法に慣れるための簡単なプログラム例を以下に示す.

```
10 INIT : CLS 4
20 FOR I=1 TO 3
30    P=I*20
40    Q=P+20
50    LINE (P,P)-(Q,Q),PSET,7,BF
60 NEXT I
70 CLS : PRINT "ｼﾛｲ ﾊｺｦ 3ｺ ｶｷﾏｼﾀ"
80 PAUSE 30
90 CLS 4
100 FOR I=1 TO 3
110     SCREEN 0,0,I
120     P=I*20
130     Q=P+20
140     LINE (P,P)-(Q,Q),PSET,7,BF
150 NEXT I
160 CLS : PRINT "ｶﾞﾒﾝ 1,2,3ﾆ ﾜｹﾃｶｸﾄ ｺｳﾅﾘﾏｽ"
170 PRINT "ｲﾛﾊﾞﾝｺﾞｳﾊ ﾄﾞﾚﾓ 7 ﾃﾞｽｶﾞ ｺﾉﾖｳﾆ ｲﾛｶﾞﾂｷﾏｽ"
180 PAUSE 70
190 CANVAS 3,5,6
200 CLS  : PRINT "CANVAS 3,5,6 ﾃﾞ"
210 PRINT "ｲﾛｦ ｶｴﾏｼﾀ"
220 PAUSE 50
230 CANVAS 2,7,1
240 CLS : PRINT "CANVAS 2,7,1 ﾃﾞ"
250 PRINT "ｲﾛｦ ｶｴﾏｼﾀ"
260 PAUSE 50
270 CANVAS 6,0,0
280 CLS : PRINT "CANVAS 6,0,0 ﾃﾞ"
290 PRINT "ｶﾞﾒﾝ 1 ﾀﾞｹｦ ﾋｮｳｼﾞ"
300 PAUSE 50
310 CANVAS 0,6,0
320 CLS : PRINT "CANVAS 0,6,0 ﾃﾞ"
330 PRINT "ｶﾞﾒﾝ 2 ﾀﾞｹｦ ﾋｮｳｼﾞ"
340 PAUSE 50
350 CANVAS 0,0,6
360 CLS : PRINT "CANVAS 0,0,6 ﾃﾞ"
370 PRINT "ｶﾞﾒﾝ 3 ﾀﾞｹｦ ﾋｮｳｼﾞ"
380 PAUSE 50
390 CANVAS 7,7,7
400 CLS : PRINT "CANVAS 7,7,7 ﾃﾞ"
410 PRINT "3 ｶﾞﾒﾝ ﾄﾓ ﾋｮｳｼﾞ"
420 PAUSE 50
430 CLS 1
440 CLS : PRINT "CLS 1 ﾃﾞ"
450 PRINT "ｶﾞﾒﾝ 1 ﾀﾞｹｦ ｼｮｳｷｮ"
460 PAUSE 50
470 CLS 2
480 CLS : PRINT "CLS 2 ﾃﾞ"
490 PRINT "ｶﾞﾒﾝ 2 ﾓ ｼｮｳｷｮ"
500 PAUSE 50
510 CLS 3
520 CLS : PRINT "CLS 3 ﾃﾞ"
530 PRINT "ｶﾞﾒﾝ 3 ﾓ ｼｮｳｷｮ"
540 PAUSE 50
550 CLS : PRINT "ｵﾜﾘ"
```

画面番号の選択　ページの選択は簡単であったが，画面番号の選択は少々やっかいである．じつは，いろいろな方法があるが，話を簡単に済ませるには，次のように説明するのがよいであろう．

書き込む画面の番号はSCREEN文に第3のパラメーターを付けて指定する*．

SCREEN　表示ページ，書き込みページ，画面番号

(例)　SCREEN 1,0,2

[**注意**]　上記の方式を用いる場合には，あらかじめ

COLOR 7,0

とし，グラフィック命令の色指定欄は（もし書くとすれば）すべて色番号7にして，要するに図形をすべて白で描くとよい．白以外の色が指定された場合，書き込む画面の本来の色（たとえば1番ならば青）以外のビットは無視される．

表示する画面の選択はCANVAS文によって行なう．これは次のように書く．

CANVAS　c_1，c_2，c_3

ただし

c_1 は第1画面の表示色番号
c_2 は第2画面の表示色番号　｝値は0～7の整数
c_3 は第3画面の表示色番号

(例)　CANVAS 0,0,7　とすれば，第1画面，第2画面の内容は表示されず，第3画面が色番号7（白）で表示される．

---

*　SCREEN文の第3パラメーターとして1～3を指定するとモノクロ書き込みになる．これを普通の（カラー7色の）書き込み可能な状態にもどすには，第3パラメーターを0にすればよい．

7.14 画面の前後関係の指定

本機では，複数画面を重ねて表示する場合，画面の前後関係を指定することができる．前後関係というのは，ある物体が，もう一つの物体の手前にあるか向う側にあるか，ということで，具体的な表示技術としては，同一の点に二つの色指定がなされたとき，どちらの色で表示するか(どちらの画面指定を優先するか)ということである．これは**LAYER**文で指定する．

LAYER　文字，画面1，画面2，画面3

└── の優先順位 ──┘

数字1，2，3，4で指定

(例)　遠景を画面1で描き，近景を画面3，その中間を画面2で描くには，画面3を最優先，次を画面2，次を画面1とすればよいから，

LAYER 1,4,3,2

と指定する（文字は，特別な事情のない限り，1にしておく）．

[簡単なプログラム例]

```
10 INIT : CLS 4
20 CANVAS 1,2,4
30 LAYER 1,2,3,4
40 FOR I=1 TO 3
50    SCREEN 0,0,I
60    P=I*20
70    Q=P+30
80    LINE (P,P)-(Q,Q),PSET,7,BF
90 NEXT I
100 CLS : PRINT "ｶﾞﾒﾝ 1,2,3ﾆ ﾊｺｦ ｶｷﾏｼﾀ"
110 PRINT "LAYER 1,2,3,4 ﾄｽﾚﾊﾞ ｺｳﾅﾘﾏｽ"
120 PAUSE 70
130 LAYER 1,3,2,4
140 CLS  : PRINT "LAYER 1,3,2,4"
150 PRINT "ﾄ ｼﾃｲｽﾚﾊﾞ ｺｳﾅﾘﾏｽ"
160 PAUSE 50
170 LAYER 1,4,3,2
180 CLS : PRINT "LAYER 1,4,3,2"
190 PRINT "ﾄ ｼﾃｲｽﾚﾊﾞ ｺｳﾅﾘﾏｽ"
200 PAUSE 50
210 LAYER 1,2,4,3
220 CLS : PRINT "LAYER 1,2,4,3"
230 PRINT "ﾄ ｼﾃｲｽﾚﾊﾞ ｺｳﾅﾘﾏｽ"
240 PAUSE 50
250 CLS 4 : PRINT "ｵﾜﾘ"
```

7.15 文字表示領域の指定

画面の使用目的には，大別して

- 文字表示（数字や文字の入出力に用いる）
- 図形表示

があり，普通は画面全体を両方に共用するわけであるが，CŌNSŌLE 文で指定すれば，文字表示領域を画面の一部分だけに制限することができる．それには次のように書く．

> CŌNSŌLE　何行目から，何行分
> ただし行番号は一番上が0で，一番下が24である．

（例）　上5行だけを文字領域に指定するのであれば，

```
CŌNSŌLE 0,5
```

本機では，横方向の領域制限（何字目から何字分を文字領域として使うという指定）も可能である．それは次のように書く．

> CŌNSŌLE　何行目から，何行分，何字目から，何字分
> ただし横方向の位置は，左端が0，右端が39（WIDTH 40の場合）または79（WIDTH 80の場合）である．

[解説]　1）スクロールは，CŌNSŌLE文で指定した範囲内で行なわれる．

2）LŌCATE文を用いれば，文字表示領域外に文字を表示することができる．ただしそのあと，カーソルを文字表示領域にもどしておかないといけない．

3）本機の場合，文字領域の指定をしなくても，文字のスクロールに伴って図形がスクロールされてしまう危険はないし，文字がじゃまなら SHIFT + CLR で文字だけを消せばよいから，普通はCŌNSŌLE文を使わなくてもよい．

7.16 ハードコピー

X1用のプリンターがあれば，画面（文字，図形とも）をプリンターでコピーすることができる．

HCŌPY　　　文字だけをコピー
HCŌPY 0　　図形だけをコピー
HCŌPY 4　　文字と図形の両方をコピー

以上が普通の表示のコピー方法である．複数画面の内の一つを選んでコピーするには，まず**SCREEN**文の第1パラメーターでページを指定し（画面に表示されている方のページがコピーされる），画面番号は次のように指定する．

HCŌPY 1　　画面1をコピー
HCŌPY 2　　画面2をコピー
HCŌPY 3　　画面3をコピー

7.17　GET@文とPUT@文

文字や図形を表示する機能の一つとして，

文字や図形のドット・パターンを配列に入れておく

それを画面上の指定位置に表示させる

ということが簡単にできると便利であろう．用途としては，たとえば

漢字を表示する

回路図の部品記号を表示する

会社や製品のシンボル・マークを表示する

模様を描く

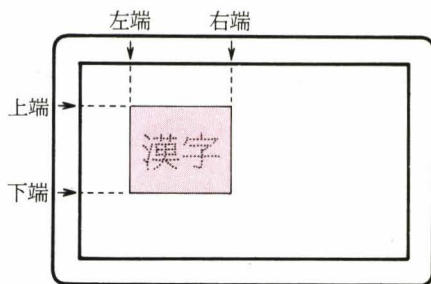
画面をディスクやテープに入れて保存する

などが考えられる．これを行なうのがGET@文とPUT@文である．その使用法には各種のバリエーションがあるので詳しい説明は省略するが，上記のような目的の場合には次のように書けばよい．

```
GET@　（左端，上端）－（右端，下端），配列名，色番号
PUT@　（左端，上端）－（右端，下端），配列名，PSET
```

ここで左端，上端，右端，下端は，ドットを読みとる区域，あるいは表示する区域で，画面座標（横座標と縦座標）で指定する．配列の型は文字列型でなければよく，要するに指定した区域のドット数，すなわち

（右端－左端＋1）×（下端－上端＋1）

に，モノクロならば1，カラー（色番号を7とする）ならば3を掛けたものを収容できるだけのビット数があればよい（実際の扱いはバイト単位だから上記の値を8で割り，端数を切り上げる）．

[使用例]　GET@文，PUT@文の簡単な応用として，画面に表示されている図形をカセット・テープに記録するプログラム，およびそれを再表示するプログラムを作ってみよう．画面全域をカラーで記録，再生するのは，記憶容量や所要時間の点で大変なので，図形は単色(色番号IRŌは1，2または4とする)，範囲は　**XMIN** $\leqq x \leqq$ **XMAX, YMIN** $\leqq y \leqq$ **YMAX**　(画面座標で指定)とする．

画面セーブのプログラム

```
8000 REM --- GET@ ---
8010 LABEL "GET"
8020 REM ｼﾃｲ ｽﾍﾞｷ ﾊﾟﾗﾒｰﾀｰ
8030 REM xmin,xmax,ymin,ymax,iro
8040 REM ﾀﾀﾞｼ iro=1,2,ﾏﾀﾊ4
8050 NX=XMAX-XMIN+1
8060 NY=YMAX-YMIN+1
8070 NW=INT((NX*NY+7)/8)/2+1
8080 DIM G%(NW)
8090 GET@(XMIN,YMIN)-(XMAX,YMAX),G%,IRO
8100 INPUT "ﾌｧｲﾙﾒｲﾊ?",F$
8110 OPEN "O",#1,F$
8120 FOR I=1 TO NW
8130 PRINT #1,G%(I);",";
8140 NEXT I
8150 CLOSE #1
8160 RETURN
```

画面ロードのプログラム

```
8170 REM --- PUT@ ---
8180 LABEL "PUT"
8190 REM ｼﾃｲ ｽﾍﾞｷ ﾊﾟﾗﾒｰﾀｰ
8200 REM xmin,xmax,ymin,ymax,iro
8210 REM ﾀﾀﾞｼ iro=1,2,ﾏﾀﾊ4
8220 NX=XMAX-XMIN+1
8230 NY=YMAX-YMIN+1
8240 NB=INT((NX*NY+7)/8)/2+1
8250 DIM G%(NW)
8260 LABEL "PUT2"
8270 INPUT "ﾌｧｲﾙﾒｲﾊ?",F$
8280 OPEN "I",#1,F$
8290 FOR I=1 TO NW
8300 INPUT #1,G%(I)
8310 NEXT I
8320 CLOSE #1
8330 PUT@(XMIN,YMIN)-(XMAX,YMAX),G%,PSET,6
8340 RETURN
```

メイン・プログラムの例（**ARRŌW**は138ページ参照）

```
10 X1=10 : Y1=10
20 X2=50 : Y2=30
30 SCREEN 0,0,0
40 COLOR 4 : CLS 4
50 GOSUB "ARROW"
60 XMIN=X1 : YMIN=Y1
70 XMAX=X2 : YMAX=Y2
80 IRO=4
90 GOSUB "GET"
100 REW
110 CLS 4
120 GOSUB "PUT2"
130 END
```

7.18 応 用 例 (挿絵参照)

例題 1 ——— 月食

月食をアニメ風に表示するプログラムを作れ.

[解答] いろいろな方法が考えられるが, 以下に示すプログラムは, **SCREEN**文によるマルチ・ページ機能を用いて書いてある. 月食の場合, 表示する円 (月) と隠す円 (地球) の半径比がかなり大きいが, 実際のとおりにすると「何も見えない時間」が長すぎてぐあいが悪いので, このプログラムではやや小さな比率を用いている.

```
10 REM --- LUNAR ECLIPSE ---
20 INIT
30 CLS 4
40 PALET 5,0
50 CIRCLE (160,100),35,6
60 PAINT (160,100),6,6
70 FOR X=60 TO 260 STEP 20
80 SCREEN 0,1,0
90 CLS 4
100 CIRCLE (160,100),35,6
110 PAINT (160,100),6,6
120 CIRCLE (X,100),60,5
130 LINE (0,0)-(319,199),PSET,5,B
140 PAINT (X,100),5,5
150 X=X+10
160 SCREEN 1,0,0
170 CLS 4
180 CIRCLE (160,100),35,6
190 PAINT (160,100),6,6
200 CIRCLE (X,100),60,5
210 LINE (0,0)-(319,199),PSET,5,B
220 PAINT (X,100),5,5
230 NEXT X
240 SCREEN 0,1,0
250 CLS 0
260 CIRCLE (160,100),35,6
270 PAINT (160,100),6,6
280 SCREEN 1,1,0
290 END
```

例題 2 ——— 国旗

世界各国の国旗をディスプレイするプログラムを作れ.

[解答]　国旗のディスプレイはグラフィック関係の命令を習得するための格好の題材である. 百ぐらい国があって, フランスのような簡単なものから, ブラジルのように, とても作れそうもないようなものまで, いろいろな難易度のものがあるが, 色は原色のものが多く, パソコンで表示するには好都合である.

以下には, 比較的作り易いもの35ヶ国の旗を作ってみた例を示す. アメリカ, イギリス, 中国など, 結構手間のかかるものがあり, これだけ作るのにまる2日（約10時間）使った. 読者はこれを全部作る必要はない. 自分の実力に合ったものを選んで作ってみればよいと思う. また, ソ連をはじめ, いくつかの重要な国が入っていないから, 自分で工夫して作ってみるとよい.

このプログラムを**RUN**させると, ただちに入力要求（?印）が出る. ここで, 表示したい国名をカナ文字で入力する（たとえば日本ならば

　　ニホン ↵

とする). その綴りは文字列型変数**I$**に読み込まれ, それがラベルとして**GŌSUB**文の行先指定に用いられ, 国名のラベルの付いたサブルーチン（たとえば日本ならば行120から始まる"**ニホン**"）が呼び出される. 国旗の表示が済むと行60にもどり, 約10秒間表示した後, 画面を消して次の国の処理に進む. 表示している間, 何もしないのではつまらないので, 次章で説明する**PLAY**文によって, ユダス･マカベウスの得賞歌を演奏することにした（行60, 70).

いくつもの国に共通の処理は, サブルーチンで処理する. それらは, プログラムの最後にまとめてある. 星形は7.9節で説明した方法によって表示している.

ニュージーランドの旗は左上にイギリスの旗が入っているので, **WINDŌW**文を活用して, うまく処理している. リビアについても同様である（**WINDŌW**文を用いてトルコ国旗を縮小表示して利用).

```
10 WIDTH 40
20 INIT
30 INPUT I$ : CLS
40 IF I$="ｵﾜﾘ" THEN END
50 GOSUB I$
60 PLAY "V8O4G7E6F3G7C7D3EFGF5ED8R5"
70 PLAY "E3FGAG5GO5C7O4GF5ED6C3C8R5"
80 CLS 0
90 INIT
100 BEEP
110 GOTO 30
120 LABEL "ﾆﾎﾝ"
130 LINE(0,0)-(319,199),PSET,7,BF
140 CIRCLE(160,100),60,2
150 PAINT(160,100),2,2
160 RETURN
170 LABEL "ﾌﾗﾝｽ"
180 L=1 : M=7 : N=2
190 GOSUB "ﾀﾃ3"
200 RETURN
210 LABEL "ｲﾀﾘｰ"
220 CLS 0
230 L=4 : M=7 : N=2
240 GOSUB "ﾀﾃ3"
250 RETURN
260 LABEL "ﾍﾞﾙｷﾞｰ"
270 L=0 : M=6 : N=2
280 GOSUB "ﾀﾃ3"
290 LINE (0,0)-(A,199),PSET,7,B
300 RETURN
310 LABEL "ﾄﾞｲﾂ"
320 L=0 : M=2 : N=6
330 GOSUB "ﾖｺ3"
340 LINE (0,0)-(319,A),PSET,7,B
350 RETURN
360 LABEL "ｵﾗﾝﾀﾞ"
370 L=2 : M=7 : N=1
380 GOSUB "ﾖｺ3"
390 RETURN
400 LABEL "ｵｰｽﾄﾘｱ"
410 L=2 : M=7 : N=2
420 GOSUB "ﾖｺ3"
430 RETURN
440 LABEL "ﾊﾝｶﾞﾘｰ"
450 L=2 : M=7 : N=4
460 GOSUB "ﾖｺ3"
470 RETURN
```

```
480 LABEL "イギリス"
490 LINE (0,0)-(319,199),PSET,1,BF
500 D=319/9
510 FOR I=-D TO D
520    LINE(I,0)-(319+I,199),PSET,7
530    LINE(I,199)-(319+I,0),PSET,7
540 NEXT I
550 D=25
560 FOR I=-D TO +D
570    LINE(I,0)-(319+I,199),PSET,2
580    LINE(I,199)-(319+I,0),PSET,2
590 NEXT I
600 E=70 : F=199-E
610 LINE (0,E)-(319,F),PSET,7,BF
620 G=319*.4 : H=319-F
630 LINE (G,0)-(H,199),PSET,7,BF
640 E=80 : F=199-E
650 LINE (0,E)-(319,F),PSET,2,BF
660 G=137 : H=319-G
670 LINE (G,0)-(H,199),PSET,2,BF
680 RETURN
690 LABEL "チリ"
700 LINE (0,0)-(319,100),PSET,7,BF
710 LINE (0,101)-(319,199),PSET,2,BF
720 LINE (0,0)-(100,100),PSET,1,B
730 POLY (50,50),25,1,144,90,810
740 PAINT (1,1),1,1
750 POLY (50,50),25,7,144,90,810
760 RETURN
770 LABEL "デンマーク"
780 LINE (0,0)-(319,199),PSET,2,BF
790 LINE (0,83)-(319,117),PSET,7,BF
800 LINE (83,0)-(117,199),PSET,7,BF
810 RETURN
820 LABEL "スイス"
830 LINE (0,0)-(319,199),PSET,2,BF
840 LINE (90,80)-(230,120),PSET,7,BF
850 LINE (140,30)-(180,170),PSET,7,BF
860 RETURN
870 LABEL "スウェーデン"
880 LINE (0,0)-(319,199),PSET,5,BF
890 LINE (0,83)-(319,117),PSET,6,BF
900 LINE (83,0)-(117,199),PSET,6,BF
910 RETURN
920 LABEL "ギリシャ"
930 LINE (0,0)-(319,199),PSET,1,BF
```

```
940 LINE (0,80)-(320,120),PSET,7,BF
950 LINE (140,0)-(180,199),PSET,7,BF
960 RETURN
970 LABEL "ﾉﾙｳｪｰ"
980 LINE (0,0)-(319,199),PSET,2,BF
990 LINE (0,84)-(319,116),PSET,7,BF
1000 LINE (84,0)-(116,199),PSET,7,BF
1010 LINE (0,88)-(319,112),PSET,1,BF
1020 LINE (88,0)-(112,199),PSET,1,BF
1030 RETURN
1040 LABEL "ｱｲｽﾗﾝﾄﾞ"
1050 LINE (0,0)-(319,199),PSET,1,BF
1060 LINE (0,84)-(319,116),PSET,7,BF
1070 LINE (84,0)-(116,199),PSET,7,BF
1080 LINE (0,88)-(319,112),PSET,2,BF
1090 LINE (88,0)-(112,199),PSET,2,BF
1100 RETURN
1110 LABEL "ﾓﾅｺ"
1120 LABEL "ｲﾝﾄﾞﾈｼﾔ"
1130 LINE (0,0)-(319,99),PSET,2,BF
1140 LINE (0,100)-(319,199),PSET,7,BF
1150 RETURN
1160 LABEL "ｴﾁｵﾋﾟﾔ"
1170 L=4 : M=6 : N=2
1180 GOSUB "ﾖｺ3"
1190 RETURN
1200 LABEL "ﾎﾟｰﾗﾝﾄﾞ"
1210 LINE (0,0)-(319,99),PSET,7,BF
1220 LINE (0,100)-(319,199),PSET,2,BF
1230 RETURN
1240 LABEL "ｱﾒﾘｶ"
1250 L=2 : M=7 : N=13
1260 GOSUB "ﾖｺｼﾏ"
1270 LINE (0,0)-(141,107),PSET,1,BF
1280 DX=142/11 : DY=108/9
1290 HX=DX/2   : HY=DY/2
1300 FOR I=1 TO 9 STEP 2
1310    Y=I*DY-HY
1320    FOR J=1 TO 11 STEP 2
1330       X=J*DX-HX
1340       POLY (X,Y),3,7,144,90,810
1350    NEXT J
1360 NEXT I
1370 FOR I=2 TO 8 STEP 2
1380    Y=I*DY-HY
1390    FOR J=2 TO 10 STEP 2
```

```
1400        X=J*DX-HX
1410        POLY (X,Y),3,7,144,90,810
1420    NEXT J
1430    NEXT I
1440 RETURN
1450 LABEL "ﾁｪｺ"
1460 LABEL "ﾁｪｺｽﾛｳﾞｧｷｱ"
1470 L=7 : M=2 : N=2
1480 GOSUB "ﾖｺｼﾏ"
1490 LINE (0,0)-(160,100),PSET,1
1500 LINE (0,0)-(0,199),PSET,1
1510 LINE (160,100)-(0,199),PSET,1
1520 PAINT (1,2),1,1
1530 RETURN
1540 LABEL "ｷｭｰﾊﾞ"
1550 L=5 : M=7 : N=5
1560 GOSUB "ﾖｺｼﾏ"
1570 LINE (0,0)-(160,100),PSET,2
1580 LINE (0,0)-(0,199),PSET,2
1590 LINE (160,100)-(0,199),PSET,2
1600 PAINT (1,2),2,2
1610 POLY (50,100),35,7,144,90,810
1620 PAINT (50,100),7,7
1630 FOR I=1 TO 5
1640    C=2*π/5
1650    X=50+30*COS(I*C+π/2)
1660    Y=100-30*SIN(I*C+π/2)
1670    PAINT (X,Y),7,7
1680 NEXT I
1690 RETURN
1700 LABEL "ﾕｰｺﾞｽﾗｳﾞｨｱ"
1710 LABEL "ﾕｰｺﾞ"
1720 L=1 : M=7 : N=2
1730 GOSUB "ﾖｺ3"
1740 POLY (160,100),80,6,144,90,810
1750 PAINT (160,100),2,6
1760 FOR I=1 TO 5
1770    C=2*π/5
1780    X=160+30*COS(I*C+π/2)
1790    Y=100-30*SIN(I*C+π/2)
1800    PAINT (X,Y),2,6
1810 NEXT I
1820 FOR R=29 TO 31
1830    POLY (160,100),R,2,144,90-36,810
1840 NEXT R
1850 RETURN
```

```
1860 LABEL "ﾘﾍﾞﾘｱ"
1870 L=2 : M=7 : N=11
1880 GOSUB "ﾖｺｼﾏ"
1890 LINE (0,0)-(91,91),PSET,1,BF
1900 POLY (45,45),35,7,144,90,810
1910 PAINT (45,45),7,7
1920 FOR I=1 TO 5
1930     C=2*π/5
1940     X=45+30*COS(I*C+π/2)
1950     Y=45-30*SIN(I*C+π/2)
1960     PAINT (X,Y),7,7
1970 NEXT I
1980 RETURN
1990 LABEL "ﾊﾟﾅﾏ"
2000 X1=0 : X2=160 : X3=320
2010 Y1=0 : Y2=100 : Y3=200
2020 LINE (X1,Y1)-(X2,Y2),PSET,7,BF
2030 LINE (X2,Y1)-(X3,Y2),PSET,2,BF
2040 LINE (X1,Y2)-(X2,Y3),PSET,1,BF
2050 LINE (X2,Y2)-(X3,Y3),PSET,7,BF
2060 POLY (80,50),35,1,144,90,810
2070 PAINT (80,50),1,1
2080 FOR I=1 TO 5
2090     C=2*π/5
2100     X=80+30*COS(I*C+π/2)
2110     Y=50-30*SIN(I*C+π/2)
2120     PAINT (X,Y),1,1
2130 NEXT I
2140 POLY (240,150),35,2,144,90,810
2150 PAINT (240,150),2,2
2160 FOR I=1 TO 5
2170     C=2*π/5
2180     X=240+30*COS(I*C+π/2)
2190     Y=150-30*SIN(I*C+π/2)
2200     PAINT (X,Y),2,2
2210 NEXT I
2220 RETURN
2230 LABEL "ﾆｭｰｼﾞｰﾗﾝﾄﾞ"
2240 WINDOW (0,0)-(159,99),(0,0)-(319,199)
2250 GOSUB "ｲｷﾞﾘｽ"
2260 WINDOW (0,0)-(319,199),(0,0)-(319,199)
2270 X1=0 : X2=160 : X3=320
2280 Y1=0 : Y2=100 : Y3=200
2290 LINE (X2,Y1)-(X3,Y2),PSET,1,BF
2300 LINE (X1,Y2)-(X2,Y3),PSET,1,BF
2310 LINE (X2,Y2)-(X3,Y3),PSET,1,BF
```

```
2320 X0=240 : Y0=40 : R=12 : IRO=2
2330 GOSUB "ホシ"
2340 X0=200 : Y0=90 : R=12
2350 GOSUB "ホシ"
2360 X0=290 : Y0=80 : R=10
2370 GOSUB "ホシ"
2380 X0=240 : Y0=170 : R=15
2390 GOSUB "ホシ"
2400 RETURN
2410 LABEL "アラブレンゴウ"
2420 L=2 : M=7 : N=0
2430 GOSUB "ヨコ3"
2440 LINE (0,B)-(319,C),PSET,7,B
2450 X0=115 : Y0=100 : R=30 : IRO=4
2460 GOSUB "ホシ"
2470 X0=205 : Y0=100 : R=30 : IRO=4
2480 GOSUB "ホシ"
2490 RETURN
2500 LABEL "イエーメン"
2510 L=2 : M=7 : N=0
2520 GOSUB "ヨコ3"
2530 LINE (0,B)-(319,C),PSET,7,B
2540 X0=160 : Y0=100 : R=30 : IRO=4
2550 GOSUB "ホシ"
2560 RETURN
2570 LABEL "イラク"
2580 L=2 : M=7 : N=0
2590 GOSUB "ヨコ3"
2600 LINE (0,B)-(319,C),PSET,7,B
2610 X0=80 : Y0=100 : R=30 : IRO=4
2620 GOSUB "ホシ"
2630 X0=160 : Y0=100 : R=30 : IRO=4
2640 GOSUB "ホシ"
2650 X0=240 : Y0=100 : R=30 : IRO=4
2660 GOSUB "ホシ"
2670 RETURN
2680 LABEL "トルコ"
2690 LINE (0,0)-(320,200),PSET,2,BF
2700 CIRCLE (106,100),53,7
2710 PAINT (106,100),7,7
2720 CIRCLE (120,100),45,2
2730 PAINT (120,100),2,2
2740 X0=187 : Y0=100 : R=27 : IRO=7
2750 POLY (X0,Y0),R,IRO,144,180,900
2760 PAINT (X0,Y0),IRO,IRO
2770 FOR I=1 TO 5
```

```
2780     C=2*π/5
2790     X=X0+.5*R*COS(I*C+π)
2800     Y=Y0-.5*R*SIN(I*C+π)
2810     PAINT (X,Y),IRO,IRO
2820 NEXT I
2830 RETURN
2840 LABEL "リビヤ"
2850 PALET 1,2
2860 PALET 2,0
2870 LINE (0,0)-(320,50),PSET,1,BF
2880 LINE (0,150)-(320,200),PSET,4,BF
2890 LINE (0,51)-(0,149),PSET,7
2900 LINE (319,51)-(319,149),PSET,7
2910 WINDOW (80,51)-(240,149),(0,0)-(319,199)
2920 GOSUB "トルコ"
2930 WINDOW
2940 RETURN
2950 LABEL "イスラエル"
2960 DY=100/6
2970 LINE (0,0)-(319,199),PSET,7,BF
2980 LINE (0,DY)-(319,50),PSET,1,BF
2990 LINE (0,150)-(319,199-DY),PSET,1,BF
3000 FOR R=43 TO 47
3010     POLY (160,100),R,1,120,30,390
3020     POLY (160,100),R,1,120,90,450
3030 NEXT R
3040 RETURN
3050 LABEL "チュウゴク"
3060 LINE (0,0)-(319,199),PSET,2,BF
3070 X0=55 : Y0=55 : R=40 : IRO=6
3080 GOSUB "ホシ"
3090 R=10
3100 FOR T=0 TO 3
3110     X0=53+60*COS(π/4-T*π/6)
3120     Y0=53-60*SIN(π/4-T*π/6)
3130     GOSUB "ホシ"
3140 NEXT T
3150 RETURN
3160 LABEL "ビルマ"
3170 LINE (0,0)-(319,199),PSET,2,BF
3180 LINE (0,0)-(142,100),PSET,1,BF
3190 X0=71 : Y0=50 : R=25 : IRO=7
3200 GOSUB "ホシ"
3210 R=8
3220 FOR T=1 TO 5
3230     THETA=2*π*(T/5+1/8)
```

```
3240     X0=71+35*COS(THETA)
3250     Y0=50-35*SIN(THETA)
3260     GOSUB "ホシ"
3270 NEXT T
3280 RETURN
7000 LABEL "ホシ"
7010 POLY (X0,Y0),R,IRO,144,90,810
7020 PAINT (X0,Y0),IRO,IRO
7030 FOR I=1 TO 5
7040     C=2*π/5
7050     X=X0+.5*R*COS(I*C+π/2)
7060     Y=Y0-.5*R*SIN(I*C+π/2)
7070     PAINT (X,Y),IRO,IRO
7080 NEXT I
7090 RETURN
7100 LABEL "タテ3"
7110 A=319/3 : B=319*2/3 : C=319
7120 LINE(0,0)-(A,199),PSET,L,BF
7130 LINE(A,0)-(B,199),PSET,M,BF
7140 LINE(B,0)-(C,199),PSET,N,BF
7150 RETURN
7160 LABEL "ヨコ3"
7170 A=199/3 : B=199*2/3 : C=199
7180 LINE(0,0)-(319,A),PSET,L,BF
7190 LINE(0,A)-(319,B),PSET,M,BF
7200 LINE(0,B)-(319,C),PSET,N,BF
7210 RETURN
7220 LABEL "ヨコシマ"
7230 DY=200/N
7240 FOR I=1 TO N
7250     Y1=(I-1)*DY
7260     Y2=I*DY
7270     IF (I MOD 2)=1 THEN IRO=L ELSE IRO=M
7280     LINE (0,Y1)-(319,Y2),PSET,IRO,BF
7290 NEXT I
7300 RETURN
```

例題 3 ―― **道路地図**

道路地図を描くプログラムを作れ.

[解答] 道路を1本の線で描くのなら簡単であるが，できれば幅を付けて描きたい．それも，道路の幅員に合わせて，広い道は広く，狭い道は狭く描きたい．一方，入力データーはなるべく簡単にしたい.

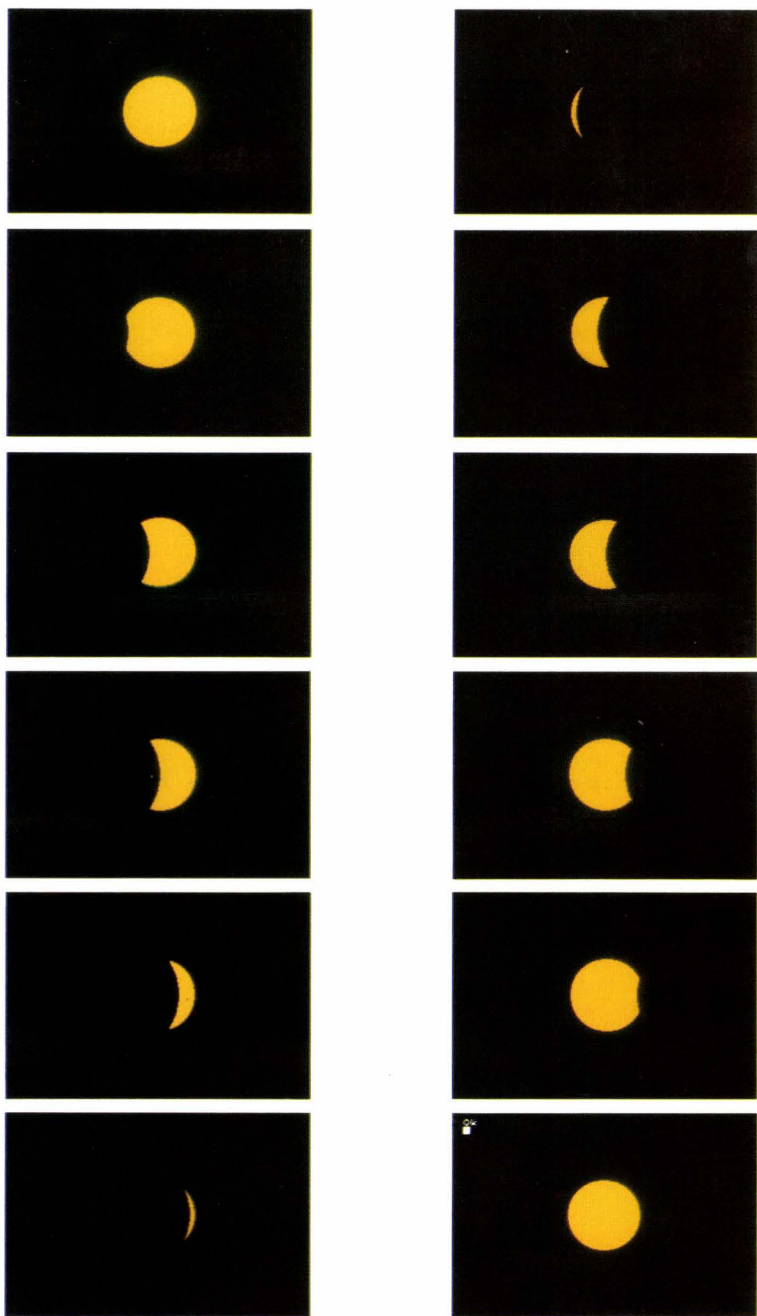
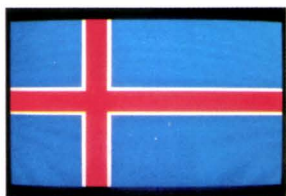
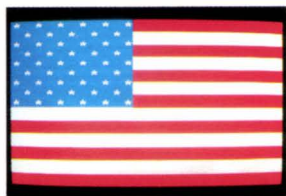
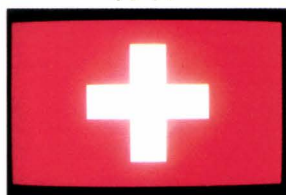
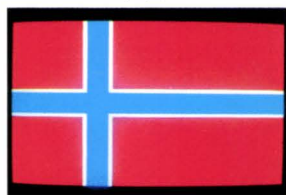
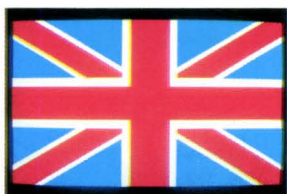
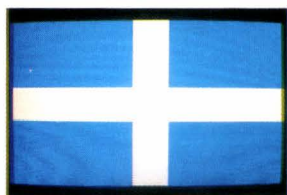
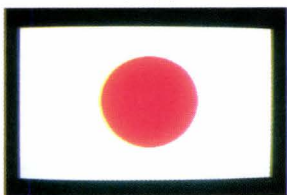
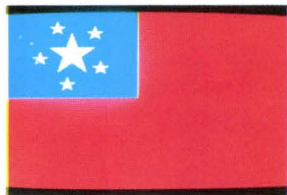
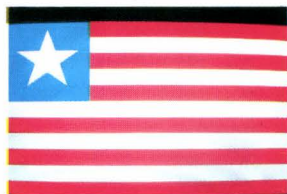
これは簡単そうに見えて，じつは結構やっかいな問題であって，正攻法でいくと，連立方程式をいくつも解かなければならないことになる．X–Y プロッターで描かせる場合にはそうする以外に手がないのであるが，パソコンのディスプレイに表示する場合には，非常に簡単な方法がある．その要点は

LINE文で道をぬりつぶす

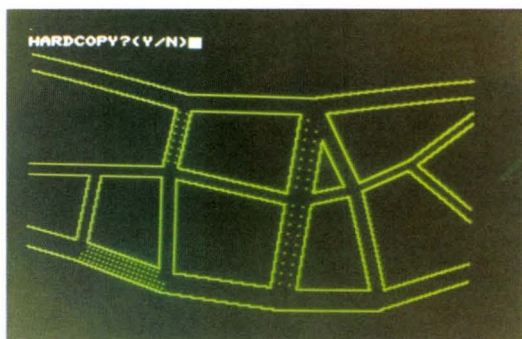
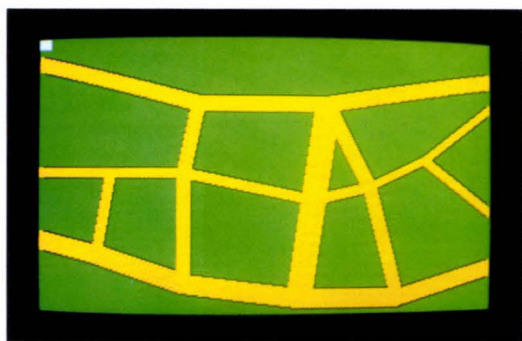
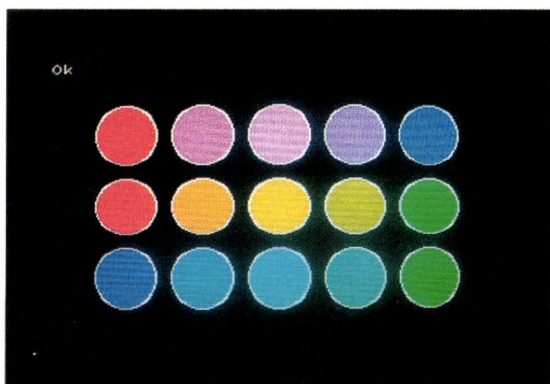
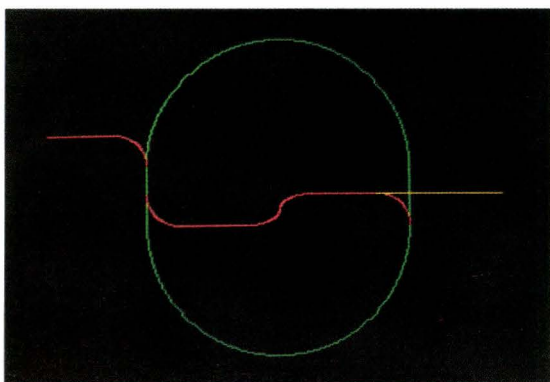
あとでその輪郭線を抽出する

ということで，後半の処理に少し時間がかかるが，ここは機械語で書けば容易に高速化できるので，原理的なところをBASICで書くと次のようになる．行460で用いているPŌINTという関数は，画面座標を与えるとそこに表示されている点の色番号を教えてくれる関数である.

```
10 REM ------- ﾁｽﾞｦｶｸ -------
20 DEFINT I-N
30 DEFINT P,Q,R
40 CLS 4
50 REM ﾃﾞｰﾀｰ ｦ ﾖﾑ
60 READ NP
70 DIM X(NP),Y(NP)
80 FOR L=1 TO NP
90     READ X(L),Y(L)
100 NEXT L
110 READ NE
120 DIM I(NE),J(NE),W(NE)
130 FOR L=1 TO NE
140     READ I(L),J(L),W(L)
150 NEXT L
160 REM ﾋｮｳｼﾞｽﾙ
170 INIT
180 WIDTH 40
190 CLS 4
200 LINE (0,0)-(319,199),PSET,1,BF
```

```
210 FOR L=1 TO NE
220     XI=X(I(L)) : YI=Y(I(L))
230     XJ=X(J(L)) : YJ=Y(J(L))
240     DX=XJ-XI
250     DY=YJ-YI
260     EL=SQR(DX*DX+DY*DY)
270     C=DX/EL
280     S=DY/EL
290     FOR U=-W(L) TO W(L) STEP .25
300         X0=XI-U*S
310         X1=XJ-U*S
320         Y0=199-(YI+U*C)
330         Y1=199-(YJ+U*C)
340         LINE (X0,Y0)-(X1,Y1),PSET,2
350     NEXT U
360 NEXT L
370 REM --- PART 2 ---
380 DIM Q(200),P(200),R(200)
390 FOR J=1 TO 199
400  P(J)=POINT(2,J)
410  R(J)=POINT(3,J)
420 NEXT J
430 FOR I=3 TO 317
440  FOR J=1 TO 199
450   Q(J)=P(J) : P(J)=R(J)
460   R(J)=POINT(I+1,J)
470  NEXT J
480  FOR J=3 TO 197
490   IF P(J)=2 GOTO "NJ"
500   IF P(J-1)=2 THEN PSET(I,J,4) : GOTO "NJ"
510   IF P(J+1)=2 THEN PSET(I,J,4) : GOTO "NJ"
520   IF Q(J)=2 THEN PSET(I,J,4) : GOTO "NJ"
530   IF R(J)=2 THEN PSET(I,J,4) : GOTO "NJ"
540   IF Q(J-1)=2 THEN PSET(I,J,4) : GOTO "NJ"
550   IF Q(J+1)=2 THEN PSET(I,J,4) : GOTO "NJ"
560   IF R(J-1)=2 THEN PSET(I,J,4) : GOTO "NJ"
570   IF R(J+1)=2 THEN PSET(I,J,4) : GOTO "NJ"
580  LABEL "NJ" : NEXT
590 NEXT I
600 PALET 1,0
610 PALET 2,0
620 LOCATE 0,0
630 INPUT "HARDCOPY?(Y/N)",I$
640 IF I$="Y" THEN HCOPY 3
650 END
```

例題1　月　食

アイスランド
アメリカ
イタリア
オランダ
スイス
チェコスロヴァキア
ドイツ
トルコ
ノルウェー
パナマ
フランス
ポーランド

例題2　国　旗

例題2　国　旗

例題3　道路地図

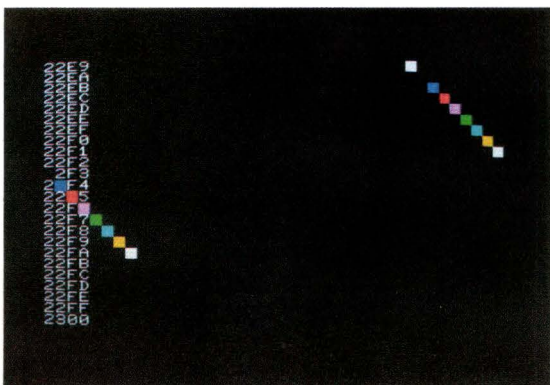
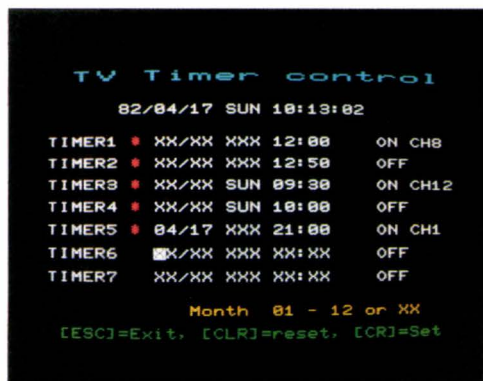
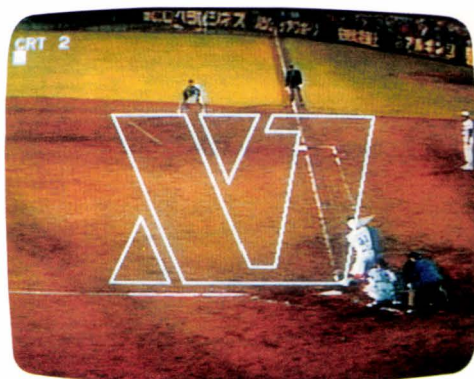
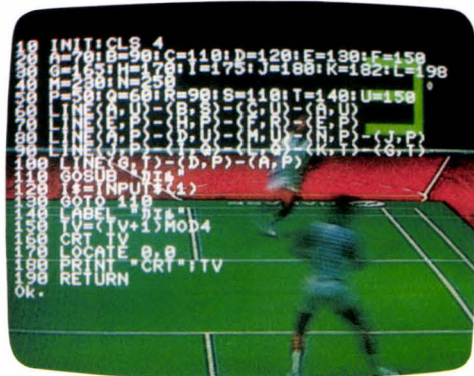
第１部終了時の画面を
PALET 2,7
により色を変えて表示したもの

輪郭線抽出の結果

実行終了後，キーボードより
PALET 1,4:PALET 2,6:PALET 4,0
を指令して着色した図

タイリングによる中間色の表示例(7.12節)

東京の国電の地図(7.10節)

POKE@の使用例(付録Ｉ)

```
TV Timer control

82/04/17 SUN 10:13:02

TIMER1 * XX/XX XXX 12:00    ON CH8
TIMER2 * XX/XX XXX 12:50    OFF
TIMER3 * XX/XX SUN 09:30    ON CH12
TIMER4 * XX/XX SUN 10:00    OFF
TIMER5 * 04/17 XXX 21:00    ON CH1
TIMER6   XX/XX XXX XX:XX    OFF
TIMER7   XX/XX XXX XX:XX    OFF

Month 01 - 12 or XX
[ESC]=Exit, [CLR]=reset, [CR]=Set
```

予約タイマー設定時の画面(付録E.2)

スーパーインポーズの例(付録E.3)

上図の表示に用いたプログラム

```
660 REM テスト・データー
670 REM NP
680 DATA 18
690 REM X(L),Y(L)
700 DATA 0,50
710 DATA 40,40
720 DATA 100,20
730 DATA 180,10
740 DATA 250,10
750 DATA 320,30
760 DATA 0,100
770 DATA 50,100
780 DATA 100,100
790 DATA 190,80
800 DATA 230,90
810 DATA 270,110
820 DATA 320,70
830 DATA 320,150
840 DATA 0,180
850 DATA 110,150
860 DATA 200,150
870 DATA 320,170
880 REM NE
890 DATA 22
900 REM I(L),J(L),W(L)
910 DATA 1,2,6
920 DATA 2,3,6
930 DATA 3,4,6
940 DATA 4,5,6
950 DATA 5,6,5
960 DATA 2,8,3
970 DATA 3,9,4
980 DATA 4,10,8
990 DATA 5,11,4
1000 DATA 13,12,3
1010 DATA 7,8,3
1020 DATA 8,9,3
1030 DATA 9,10,3
1040 DATA 10,11,3
1050 DATA 11,12,2
1060 DATA 12,14,2
1070 DATA 9,16,4
1080 DATA 10,17,8
1090 DATA 11,17,4
1100 DATA 15,16,5
1110 DATA 16,17,5
1120 DATA 17,18,5
```

(91ページからの続き)

```
290 GOSUB "ｶﾝｻﾝ"
300 R=N-INT(N/7)*7
310 IF R<=1 THEN KARA=-4-R ELSE KARA=3-R
320 REPEAT
330    MADE=KARA+6
340    IF MADE>D(M) THEN MADE=D(M)
350    FOR i=KARA TO MADE
360       COLOR 7
370       IF I=KARA THEN COLOR 2
380       IF I=KARA+6 THEN COLOR 1
390       IF I<=0 THEN PRINT SPC(6);
                         ELSE PRINT USING " ##   ";I;
400    NEXT I
410    PRINT
420    KARA=KARA+7
430 UNTIL KARA>D(M)
440 COLOR 4
450 PRINT : PRINT
460 END
470 REM --- 1600.1.0 ｶﾗﾉ ﾋｽｳﾆ ｶﾝｻﾝ ---
480 LABEL "ｶﾝｻﾝ"
490 Z=Y-1600
500 N2=365*Z+Z¥4-Z¥100+Z¥400
510 IF Z>0 AND D(2)<>29 THEN N2=N2+1
520 N1=C(M)+1
530 N=N1+N2
540 RETURN
```

```
年,月 ? 1983,7


              1983  年 7 月

  日    月    火    水    木    金    土
                                 1     2
  3     4     5     6     7     8     9
 10    11    12    13    14    15    16
 17    18    19    20    21    22    23
 24    25    26    27    28    29    30
 31

OK
```

8 音響出力

8.1 PLAY 命令の機能概説

本機は音楽演奏機能をもち，そのプログラムをBASICで書くことができる，

3声部の和声出力可能

音　域　　8オクターブ

音　量　　15段階制御可能

で，市販のパソコンの中では(お値段100万円也のヤマハ YIS を論外とすれば）高級の部類に属する．

楽譜の入力方法は，MML (Music Macro Language) に似た方式で，器楽または声楽をやっている人には非常に親しみ易い．MML は世界共通で，他のパソコンもこれに似た方式を採用しているものが多いから，覚えておけば便利である．

本論に入る前に，簡単な例題を示す．

```
10 PLAY"CDErCDErGEDCDEDr"
20 PLAY"CDErCDErGEDCDECr"
30 PLAY"GGEGAAGrEEDDC2.r"
```

これは童謡「チューリップ」を演奏するプログラムである．第1行は

で，**CDE**…は音階のドイツ読みであり，**r** は休止符を表す．3行目の **2.** は付点2分音符であることを表す接尾語である．

8.2 音程の表し方

音階は，ハ長調の

　　ド　レ　ミ　ファ　ソ　ラ　シ

を

　　C　D　E　F　G　A　B

で表す(いわゆるドイツ式の読み方に同じ．ただしHは使わない)．いいかえれば，日本式の読み方のイロハニホヘトがABCDEFに対応する．本機の半音指定方法は標準のMMLと少し異なり

　#(半音上げ)は，記号#を音名の左に付けて表す

　♭(半音下げ)は，記号#を一つ下の音名の左に付けて表す

(例)

は#F

は#A

同じCでも，中央のC，上のC，下のC，などがある．これを区別するにはŌ印による**音域指定**を左に付ける(Ōはoctaveの意)．

　上のCは　　Ō5C

　中央のCは　Ō4C　　(C以外についても同様．)

　下のCは　　Ō3C

最も低いのがŌ1，中央がŌ4，最も高いのがŌ8である．

すべての音符にこれを書くのは大変なので，次のような規則で省略することができる．

1) Ōを書かなければ「一つ前と同じ」とみなす．

2) 電源投入時にはŌ4の状態になる．ただしRUNコマンドでは初期化されない．したがって，何回も(または何曲も)演奏する場合のことを考えて，曲の最初にはなるべくŌ指定を書いておく方が安全である．

3) 現在は指定されているオクターブの「一つ上」は+印，「一つ下」は−印で表す．

(例)　Ō4のとき，+Cは「上のC」(Ō5C)，−Cは「下のC」(Ō3C)を表す．

8.3 音符の長さの指定

本機の指定方法は標準のMMLと異なり，次のようになっている．

音符		コード	音符		コード
32分音符	𝅘𝅥𝅰	0			
16分音符	𝅘𝅥𝅯	1	付点16分音符	𝅘𝅥𝅯.	2
8分音符	♪	3	付点8分音符	♪.	4
4分音符	♩	5	付点4分音符	♩.	6
2分音符	𝅗𝅥	7	付点2分音符	𝅗𝅥.	8
全音符	𝅝	9			

■ 長さの指定は音名の後(右)に付ける．

(例) C5

■ 前(左)の音と同じ長さの場合は省略してよい．したがって，同じ長さの音符がいくつも続くときは最初の音符だけに長さ指定を付ければよい．

■ 休止符はRの右に長さ指定を付けて表す．

(例) 4分休止符 𝄽 はR5　　8分休止符 𝄾 はR3

■ 演奏の速さはTEMPŌ文で次のように指定する．

TEMPŌ　1分間あたりの4分音符の数

(例) ♩ ＝70　ならば　TEMPŌ　70

8.4 PLAY文

以上で，いわば「コンピューター用の楽譜」の書き方を説明したが，その実行（演奏）はPLAY文（MUSICと書いてもよい）によって行なう．これは次のように書くく．

単音の場合

PLAY　"音符を表す文字列"

（例）　PLAY　"CDEFG"

2声部の場合

PLAY　"第1声部の文字列：第2声部の文字列"

（例）　PLAY　"C:F"

3声部の場合

PLAY　"第1声部：第2声部：第3声部"

（例）　PLAY　"CCC−BCC:EFEDEE:GAGGGG"

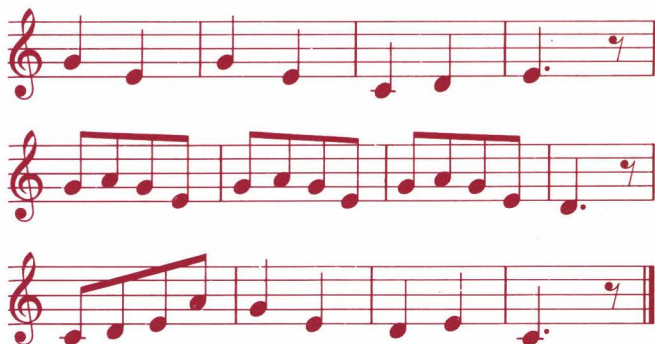
■　文字列は文字列型定数でも文字列型変数でもよい．また文字列式を書いてもよい．

■　指定は原則として各声部ごとに独立である．

■　PLAY文は普通のプログラムの順序制御規則に従って実行される．すなわち、原則として行番号順に実行され，FŌR文によるくりかえし制御も可能である．演奏機構はCPU（普通のプログラムを実行する部分）と独立になっており，PLAY文の文字列型データーを演奏機構に渡してしまうとCPUは解放されるので，PLAY文と次のPLAY文の間に若干の計算その他の処理を行なっても，音が途切れる心配はない．

8.5 応 用 例

例題 1 ―― つき

文部省唱歌の「つき」のプログラムを作ってみよう．楽譜は

［解答］

```
10 TEMPO 120
20 PLAY "G5EGECDE6R3"
30 PLAY "G3AGEGAGEGAGED6R3"
40 PLAY "C3DEAG5EDEC6R3"
```

例題 2 ―― 低い音，高い音

上記のプログラムにŌ1～Ō8の指定を付けて実行してみよう．

［解答］

```
10 TEMPO 120 : CSIZE 3 : CLS
20 FOR I=1 TO 7
30     LOCATE 18,12 : COLOR I
40     PRINT #0,"O";I
50     I$=RIGHT$(STR$(I),1)
60     PLAY "O"+I$+"G5EGECDE6R3"
70     PLAY "G3AGEGAGEGAGED6R3"
80     PLAY "C3DEAG5EDEC6R3"
90 NEXT I
```

これを聞けばわかるように，低音域はかなり音程が悪く音色もきたないからなるべく使わない方がよい．気持よく聞けるのはŌ3～Ō6である．Ō8の音も出せるが，音楽的に使える音ではない．

例題 3　雪のおどり

最近，小学生がよくタテ笛で吹いている「雪のおどり」という曲（チェコスロヴァキア民謡）のプログラムを作れ．

[解答]　同形の箇所が多いので文字列型変数を用いるとよい．

```
10 REM ------- ﾕｷﾉｵﾄﾞﾘ -------
20 A$="+D3RAR+DRARAGFGARAR"
30 B$="+D3RAR+DRARAGFEDRDR"
40 C$="F3RDRFRDRFEFGARAR"
100 PLAY A$
110 PLAY B$
120 PLAY C$
130 PLAY B$
140 PLAY C$
150 PLAY B$
```

[解説]　このプログラムは楽譜のとおりでなはく，かなり余分に休止符を入れている．本機で演奏する場合，こうしないとレガートになりすぎてしまう．

行20，行30の右辺の冒頭にある＋印は「上のD」の指定である．こういう所は普通の MML と異なるので注意されたい．

例題 4　2部輪唱

前問の曲を2部輪唱で演奏するプログラムを作れ．

[解答]　いろいろな書き方があるが，下記は比較的エレガントな解答である．**PART1$**が先発パート，**PART2$**が後発パートである．両パートを同じ音域で演奏すると区別が付ななくて面白味がないので，先発パートを1オクターブ上げてみた（これは成功で，聴いてみると割合に感じがよい）．

```
10 REM ------- ﾕｷﾉｵﾄﾞﾘ -------
20 A$="+D3RAR+DRARAGFGARAR"
30 B$="+D3RAR+DRARAGFEDRDR"
40 C$="F3RDRFRDRFEFGARAR"
50 PART1$="O5"+A$+B$+C$+B$+C$+B$+"R3RRR"
60 PART2$="O4"+"R3RRR"+A$+B$+C$+B$+C$+B$
70 PLAY PART1$+":"+PART2$
```

例題 5 ―― フレール・ジャック

フレール・ジャックを3部輪唱で演奏するプログラムを作れ.

[解答]

```
10 REM ------- ﾌﾚｰﾙ･ｼﾞｬｯｸ -------
20 A$="C5DECCDEC"
30 B$="EFGGEFGR"
40 C$="G3AGFE5CG3AGFE5C"
50 D$="C-GCRC-GCR"
60 PLAY "O4"+A$
70 PLAY B$
80 PLAY C$
90 PLAY D$
100 PLAY A$
110 PLAY B$+":"+"O5"+A$
120 PLAY C$+":"+B$
130 PLAY D$+":"+C$
140 PLAY A$+":"+D$
150 PLAY B$+":"+A$
160 PLAY C$+":"+B$+":"+"O6"+A$
170 PLAY D$+":"+C$+":"+B$
180 PLAY A$+":"+D$+":"+C$
190 PLAY B$+":"+A$+":"+D$
200 PLAY C$+":"+B$
210 PLAY D$+":"+C$
220 PLAY "O3"+A$+":"+D$
230 PLAY B$
240 PLAY C$
250 PLAY D$
260 PLAY "O3C-GCRC-GO2C7"
```

例題 6 ―― ふるさと

文部省唱歌「ふるさと」を伴奏付きで演奏するプログラムを作れ.

[解答] 伴奏の方を適当に簡略化して作った例を以下に示す.

```
10 REM ------- ﾌﾙｻﾄ -------
20 PLAY "C6R3C6R3C5CDDDEDD"+":C5-GC-GC-G-B-G-BC-B-G"
30 PLAY "E6R3E6R3F5FGGGGGR"+":C5-GC-GD-GE-GE-GE-G"
40 PLAY "FFGGAAEEEFEE"+":C5-AD-BECC-GC-GC-G"
50 PLAY "D6R3D5D-B7C5CCCCR"+":O3B5GBGBGO4C-GC-GC-G"
60 PLAY "DCDD-G7C5DE6R3E5E"+":O3BABGBGEGO4C-GC-G"
70 PLAY "FEFFFAGFEEEEEE"+":-B-A-B-G-B-GC-BC-GC-GC-G"
80 PLAY "G6R3G6R3G5GCCCDEE"+":C7-B-#A-A5-#G-A-BC-A"
90 PLAY "F6R3F5FDDCCCCCR"+":D-G-A-F-B-GC-GC-GCR"
```

例題 7 ―― ゴセックのガボット

ゴセックのガボットを演奏するプログラムを作れ.

[解答]　くりかえしが多いのでFŌR文を用いる.

```
10 TEMPO 250 : PLAY "O4"
20 A$="G5AGEFGFDCRB3+C5RC8R5"
30 B$="F5GFDEFECDR#F3G5R-G8R5"
40 C$="E5EC-ACC-A-#F-GR#F3G5R-G8R5"
50 FOR I=1 TO 2
60    PLAY A$
70    PLAY B$
80    PLAY A$
90    PLAY C$
100 NEXT I
110 D$="D5FEGFEDC-B7DF8R5"
120 E$="E5GFAGFEDC7EG8R5"
130 F$="A5G3RGRFRFRERERDRD7FA8R5"
140 G$="G5E-BCFD-A-BCRB3+C5RC8R5"
150 PLAY D$
160 PLAY E$
170 PLAY F$
180 PLAY G$
190 P$="E6R3E6R3F6R3F6R3G5+CB+CG8R5"
200 Q$="C6R3C6R3D6R3D6R3E3G#FGAGFED3R-GR-BR-GR"
210 R$="-A7C5-AAR-AR-G7C5-GGR-GR"
220 S$="F5R-GRER-GRDRD3EFED8R5"
230 FOR I=1 TO 2
240    PLAY P$
250    PLAY Q$
260    PLAY R$
270    PLAY S$
280 NEXT I
290 T$="F5RF3EDC-B7B+C5R-GRC8R5"
300 U$="F5RF3EDC-B7B+C5RCRE8R5"
310 V$="O3A7+C3BAGF7O4AGG3AGFE7G"
320 W$="F7A3GFED7B+C5RGRC8R5"
330 FOR I=1 TO 2
340    PLAY T$
350    PLAY U$
360    PLAY V$
370    PLAY W$
380 NEXT I
390 PLAY A$
400 PLAY B$
410 PLAY A$
420 PLAY C$
425 PLAY D$
430 PLAY E$
440 PLAY F$
450 PLAY G$
```

例題 8　月光の曲（一部分）

ベートーベンの月光の曲を演奏するプログラムを作れ.

[解答]　この曲は嬰ハ短調でシャープが4箇も付くので，文字列型変数に＃付きの音を入れておいて，それをつなぐことにした(行10〜230)．また8分音符の音，4分符の音，2分音符の音も用意した(行240〜380)．よく使われる伴奏パターンは最初に作っておくことにした(行400〜470)．こうした準備をして，最初の9小節を書いてみたのが以下のプログラムである(行1000〜1080の1行が1小節)．

```
10 DEFSTR A-H
20 DEFSTR O-Z
30 C="#C"
40 D="#D"
50 E="E"
60 F="#F"
70 G="#G"
80 A="A"
90 B="B"
100 CC="-#C"
110 DD="-#D"
120 EE="-E"
130 FF="-#F"
140 GG="-#G"
150 AA="-A"
160 BB="-B"
170 CCC="+#C"
180 DDD="+#D"
190 EEE="+E"
200 FFF="+#F"
210 GGG="+#G"
220 AAA="+A"
230 BBB="+B"
```

```
240 C1="O3#C8" : C2=C1+C1 : C4=C2+C2
250 D1="O3#D8" : D2=D1+D1 : D4=D2+D2
260 E1="O3E8"  : E2=E1+E1 : E4=E2+E2
270 F1="O3#F8" : F2=F1+F1 : F4=F2+F2
280 G1="O3#G8" : G2=G1+G1 : G4=G2+G2
290 A1="O3A8"  : A2=A1+A1 : A4=A2+A2
300 B1="O3B8"  : B2=B1+B1 : B4=B2+B2
310 CC1="O2#C8" : CC2=CC1+CC1 : CC4=CC2+CC2
320 DD1="O2#D8" : DD2=DD1+DD1 : DD4=DD2+DD2
330 EE1="O2E8"  : EE2=EE1+EE1 : EE4=EE2+EE2
340 FF1="O2#F8" : FF2=FF1+FF1 : FF4=FF2+FF2
350 GG1="O2#G8" : GG2=GG1+GG1 : GG4=GG2+GG2
360 AA1="O2A8"  : AA2=AA1+AA1 : AA4=AA2+AA2
370 BB1="O2B8"  : BB2=BB1+BB1 : BB4=BB2+BB2
380 BS1="O3C8"  : BS2=BS1+BS1 : BS4=BS2+BS2
390 Z=":"
400 GCE=GG+C+E : GCE2=GCE+GCE : GCE4=GCE2+GCE2
410 ACE=AA+C+E : ACE2=ACE+ACE
420 QADF=AA+"D"+F : QADF2=QADF+QADF
430 GDF=GG+D+F : GDF2=GDF+GDF : GDF4=GDF2+GDF2
440 ACF=AA+C+F : ACF2=ACF+ACF
450 GHD=GG+B+D : GHD2=GHD+GHD
460 GHE=GG+BB+E : GHE2=GHE+GHE : GHE4=GHE2+GHE2
470 AHD=AA+BB+D : AHD2=AHD+AHD
1000 PLAY GCE4+Z+C4
1010 PLAY GCE4+Z+BB4
1020 PLAY ACE2+QADF2+Z+AA2+FF2
1030 PLAY GG+"C"+F+GCE+GG+C+D+FF+"C"+D+Z+GG4
1040 PLAY EE+GG+C+GCE+GCE2+Z+C4+Z+"R8R8R8#G7R1#G3"
1050 PLAY GDF4+Z+BS4+Z+"#G8#G8R8#G7R1#G3"
1060 PLAY GCE2+ACF2+Z+C2+FF2+Z+"#G8#G8A8A8"
1070 PLAY GHE2+AHD2+Z+BB4+Z+"#G8#G8#F8B8"
1080 PLAY GHE4+Z+E4+Z+"E8R8R8R8"
```

8.6 SOUND文

本機には音響出力に関し，もう一つ，SOUNDという命令がある．これは本格的に説明すると非常に長い話になってしまうので，ここでは多少「独断と偏見」をお許しいただくことにして，ごく簡単な解説を試みる．

SOUND文の機能は大きく分けて二つある．それは

{ 音程をもつ音（楽音という）を出す
{ 音程をもたない音（雑音という）を出す

である．

この内，楽音に関する機能は，大部分，PLAY命令に含まれている．また，SOUND文で曲を書こうとする大変な手間がかかる．いうなればSOUND文は機械語で，それを音楽用に使い易くした言語がPLAY命令のMMLである．したがって，普通は，音楽演奏のためにSOUND文を使う必要はない，といってよい．

——もちろん，SOUND文を使えば，PLAY命令よりもキメの細かい制御ができる．たとえば，音程を半音間隔よりも細かく指定できるから，特定の音をほんの少しだけ低くしたいとか高くしたいとかいうような細工も可能であり，純正律に近い和音を出すことも可能である．

しかし，どんなに凝ってみても，所詮（しょせん）本物のシンセサイザーのようにはいかない．ロー・オッシレーター（低周波形発生器）による制御のできるのは音量だけである．そのため，ビブラートをかけられない．音色も限られており，低い音はブザーのような音になってしまう．まあ，そういうわけだから，本機の音楽演奏は，なるべく手軽にPLAY文で書くのが賢明であると思う．

それに対し，雑音の方はPLAY文では出せないから，SOUND文で書く必要がある．そこで「SOUND文は雑音を出す命令である」と割り切ってしまうと話は非常に簡単で，要するに次のことを覚えればよい．

SOUND 文による雑音発生の基本形　雑音を出すには,

SŌUND　6,音の高さの指定

SŌUND　7,7

SŌUND　8,音量レベル

の三つの文を実行すればよい（順序は上記のとおりでなくてもよい).

第1の文は，雑音の平均的な音の高さを指定するもので，0～31を指定できる．音の感じは，実際に聞いてみるのが最もよいが，あえて文字で書くなら，SŌUND　6,10　だと「シー」という音，SŌUND　6,20　だと「シャー」という音，SŌUND　6,30　だと「ザー」という音になる．

第2の文は「雑音スイッチ ON」という機能をもつ．

第3の文は，PLAY文のV機能と同じで，0～15を指定できる．

SŌUND　8,1　　いちばん小さな音

SŌUND　8,8　　普通の音

SŌUND　8,15　いちばん大きな音

以上の指定で出る音は連続音である．音量をあるパターンに従って制御するには次のように書けばよい,

SŌUND　6,音の高さの指定　｝上と同じ
SŌUND　7,7　　　　　　　｝

SŌUND　8,16+音量レベル

SŌUND　13,音量パターンの指定（下図参照)

SŌUND　12,音量変化の周期の指定値を256で割った商

SŌUND　11,音量変化の周期の指定値を256で割った余り

Mが小さいとピチカート
Mが大きいと鐘の音

M150ぐらいで用いると
音の区切りをつけるのに有効

M150ぐらいで用いると
トレモロの感じになる

M700ぐらいで用いると
マンドリンの感じになる

SOUND 文の文法　普通の雑音を出すだけならば以上の要領で一応，十分だと思うが，参考のため，**SŌUND**文の全機能とその指定のしかたを説明しておく，**SŌUND**文には常に二つの引数を書く．第1の引数は機能種別を表すもの，第2の引数は指定する値である．

SŌUND0～SŌUND5は楽音の周波数を指定する．その内，0と1は第1声部，2と3は第2声部，4と5は第3声部の周波数に対応する．ただし，第2引数としては周波数そのものを書くのではなく，

　　76800÷周波数

を書く．より詳しくいえば，上記の値を256で整除した商を奇数番号の命令(SŌUND1，SŌUND3，SŌUND5)の第2引数として書き，256で割って余りを偶数番号の命令の第2引数に書く．

SŌUND6は雑音周波数の指定（3声部共通）で，既に前ページで要点を説明した．

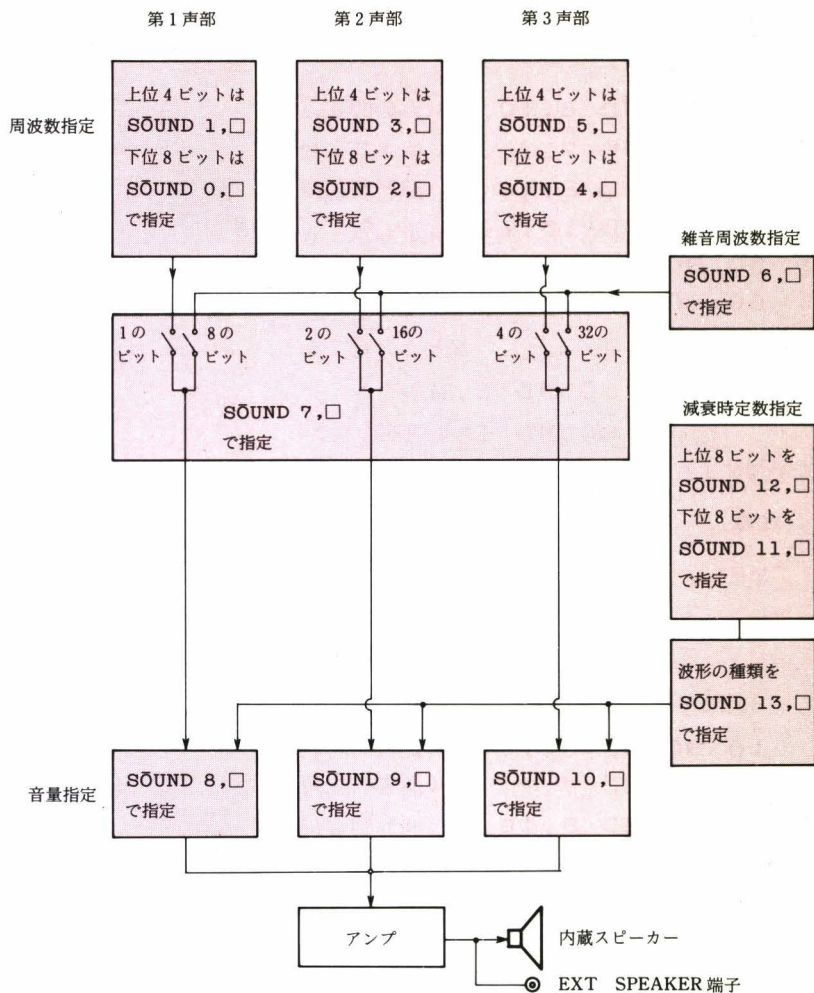
SŌUND7は楽音および雑音の，各声部独立のスイッチで，

雑音			楽音		
第3声部	第2声部	第1声部	第3声部	第2声部	第1声部
2^5	2^4	2^3	2^2	2^1	2^0

のように対応し，0がON，1がOFFを表す（普通と逆なので注意)．

(例)　第1声部の雑音と第2声部の楽音をONにしたければ，ビット・パターンは，110101であるから，10進になおせば53になり，**SŌUND 7,53**と書くことになる．

SŌUND3～SŌUND10は第1声部～第3声部の音量の指定で，さらに**SŌUND11～SŌUND13**の指定（エンベロープ指定という）を付けたい場合は，第2引数に16を加える．これは楽音に対しても雑音に対しても有効である．

第1声部
第2声部
第3声部
周波数指定
上位4ビットは
SŌUND 1,□
下位8ビットは
SŌUND 0,□
で指定
上位4ビットは
SŌUND 3,□
下位8ビットは
SŌUND 2,□
で指定
上位4ビットは
SŌUND 5,□
下位8ビットは
SŌUND 4,□
で指定
雑音周波数指定
SŌUND 6,□
で指定
1の
ビット
8の
ビット
2の
ビット
16の
ビット
4の
ビット
32の
ビット
SŌUND 7,□
で指定
減衰時定数指定
上位8ビットを
SŌUND 12,□
下位8ビットを
SŌUND 11,□
で指定
波形の種類を
SŌUND 13,□
で指定
音量指定
SŌUND 8,□
で指定
SŌUND 9,□
で指定
SŌUND 10,□
で指定
アンプ
内蔵スピーカー
EXT SPEAKER 端子

SOUND 機能概念図

[SOUND文の簡単な使用例]

1) SLの音

```
10 SŌUND 6,10      比較的高い音
20 SŌUND 7,7       決まり文句
30 SŌUND 8,16      波形制御ON
40 SŌUND 13,14     /\/\/\  形波形
50 SŌUND 12,1   }
60 SŌUND 11,0   }  比較的短い周期
```

これは快調に速く走っているときの音である.

行10の SŌUND 6,音の高さ

行50の SŌUND 12,周期の長さ

を調節すれば,いろいろな走行状態の音が出る.

[注意] このプログラムは SHIFT + BREAK では止まらない. CTRL + D とすれば止まる.プログラム的に止めるには, PAUSE文等によって適当な時間だけ待ったのちに SŌUND 8,0とすればよい.

2) ミサイルの発射音

```
10 SŌUND 6,10      比較的高い音
20 SŌUND 7,7       決まり文句
30 SŌUND 8,16      波形制御ON
40 SŌUND 13,0      |\______  形波形
50 SŌUND 12,20  }
60 SŌUND 11,0   }  中程度の時定数
```

やはり行10と行50の第2パラメーターを調節することにより,大きなミサイルのような感じから,ピストルのような感じまで,いろいろと変えることができる.

付録A　デバックの方法

A. 1　デバッグとは

新しいプログラムを作成してコンピューターに入力したとき，それが期待どおりに動いてくれればよいが，実際には最初からうまく動くとは限らず，

途中でエラー・メッセージが出て止まる

いくら待っても答が出ない

一応は結果が出るが答がおかしい

など，トラブルの起こることも少なくない．それは

プログラムの基本設計における考え落とし

プログラム作成時の文法的誤り

プログラムが正しく入力されていない

データーが正しく入力されていない

誤操作

などによるのであるが，いずれにしてもその原因をつきとめて，正しく動くように修正する必要がある．そのための作業をデバッグ(debug，虫取り）という．

デバッグはプログラム作成よりもむずかしく，時間もかかる．予想しなかったことが起こるからである．デバッグの万能的方法（こうすれば必ず原因が見付かるというような）は存在しない．しかし，これまでの経験から，いくつかの比較的有効な方法が知られている．以下では，そのような方法について簡単に紹介する．

A. 2　実行前のチェック

プログラムを入力したら，実行前にLISTコマンドによってリストを表示させ(プリンターがあればLLISTコマンドにより印刷して)，誤りがないかどうか，よく点検しておくとよい．

行番号順に目読するだけでもよいが，その際，

　　左カッコに対応する右カッコがあるか？

　　"印に対応する"印があるか？

　　FŌR文に対応するNEXT文があるか？

　　GŌTŌ文の行先は実在するか？

　　GŌSUBとRETURNは対応しているか？

というような形式的整合性をチェックしておくとよい．それは

　　この種の誤りが非常に多い

　　この種の誤りがあると原因の解明に苦労する

　　作業が簡単なわりに効果が大きい

などの理由による．

もう少し手間をかけてよければ，読み合わせをして

　　抜けている行はないか？

　　綴りが正しく入力されているか？

　　数値が正しく入力されているか？

　　数式が正しく入力されているか？

などを確認しておくとよい．この種の誤りは，文法的な誤りにならないことが多く，そのため発見が遅れて苦労するから，事前に十分チェックしておくのが賢明である．

A.3 実行中に異常停止した場合

実行中に異常停止したら，まず画面の記録をとる．プリンターがあれば画面コピーをするといい．それには，キーボードから HCŌPY 4 ↵ と指令すればよい．グラフィックなし（文字だけ）ならば HCŌPY ↵ を用いると少し時間が節約になる．

たいてい，エラー・メッセージが出ているから，それをヒントにして原因を考える．また，最後に実行した（または実行しようとした）行の番号も表示されるから，それも原因解明のヒントにする．

（例）
```
10 INPUT "X=",X,
20 Y=LŌG(X)
30 PRINT Y
```
を実行すると
```
Syntax error in 10
```
というエラー・メッセージが出て止まる．これは「行番号10の文に文法的誤りがある」という意味であるから，その行を詳しく調べると，

命令の綴り **INPUT** は正しい
入力要求 **"N="** の書き方は正しい
変数名 **X** を用いるのは正しい

ということで，残る「**,**」の文法を調べてみると，**INPUT** 文の最後がコンマではいけない，ということがわかる．

エラー・メッセージと行番号だけで解決しない場合，メモリーの内容（停止した時点における各変数の値）を調べてみるとよい．それには，**PRINT**（または **LPRINT**）文を直接実行形式で用いる．

（例） 変数 **A，B，C** の値を見たい場合
```
PRINT A;B;C
```
（例） 配列 **X** の内容（添字 **1** から **N** まで）を見たい場合
```
FŌR I=1 TŌ N:PRINT X(I):NEXT I
```
[注意] 本機ではプログラムの修正を行なっても変数の内容は消えないで残るが，**RUN** したりすればこわれるから，メモリーの内容はなるべく早く（止まったらすぐ）調べておくとよい．

A.4　主なエラー・メッセージの読み方 (ABC順)

Already open

「オープンしてあるファイルをまたオープンしようとした」

その前にファイルを使用したときのCLŌSEを忘れているためと考えられる.

Bad allocation table

「ファイルの内容一覧表がこわれている」

Bad file descripter

「ファイル名がおかしい」

たとえば"A:B"などというファイル名を使うと, このメッセージが出る.

Bad file mode

「そのテープは読めない」

テープの記録形式には, BASICのプログラム(ASCIIセーブでないもの), BASICのデーター(ASCIIセーブされたプログラムを含む), 機械語（2進法ビット・パターン）の三つがあり, 形式の異なるテープは入力できない.

Bad file number

「ファイル番号がおかしい」

ŌPENの書き忘れ, または番号違い.

Can't continue

「CŌNT指令が出たが実行再開できない」

BREAKキーを押したり, STŌP文を実行したりして一時停止したとき, コンピューターは「現在の停止位置」や停止時の状態を記憶しておくのであるが, 修正中に複雑な操作をすると, その情報がこわれてしまい, 再開できなくなる.

Device full

「ディスク満員」

この場合，満員になったディスクに既に書き込んであるプログラムやデーターの保全を第一に考え，キーボードから

CLŌSE ⏎

を入れて，しめくくっておく．書き込み途中のデーターは(まだメモリーに残っていれば）最初から全部あらためて別のディスクに書き込む．

Device in use

「指定された入出力装置が現在使用中」

Device I/O ERROR

「周辺装置の故障またはディスケットの不良」

Device offline

「指定された入出力装置が使用可能な状態にない」

スイッチの入れ忘れなどである．なお，プリンターの SEL スイッチを押し忘れてオフライン状態になっている場合には，このメッセージは出ず，コンピューターは黙って停止して，使用者が SEL スイッチを押すのを待っている．プリンターを使うプログラムで，異常に長い時間がかかる場合は SEL の押し忘れを疑うとよい．

Division by Zero

「0で割った」

除数（割る数，分母）が0になる原因はいろいろあるが，一つの可能性としては，その変数に値が代入されていなかった（未定義であった），あるいは，代入したつもりであったが綴りをまちがえていた，ということが考えられる．PRINT文を直接実行形式で用いて，関係する変数の値を調べてみるとよい．

(例)
```
10 INPUT BUNSI,BUNBŌ
20 A=BUNSI/BUNBO
30 PRINT A
```

原因はŌ（オー）と0（ゼロ）の入れまちがい．

Duplicate Definition

「配列または関数の2重定義」

たとえ寸法が同じでも，同じ配列名を2重に配列宣言することは禁止されている．そのようなプログラムを書いたはずがないのに，このメッセージが出たとすれば，次のような原因が考えられる．

ループの中にDIM文を書いてある．

GŌTŌ文などにより，DIM文より前にもどってきた．

また，GŌTŌ文の直接実行によってプログラムをスタートさせると，以前の宣言がキャンセルされていないために，2重定義ということになってしまうことがある(RUNコマンドで開始すれば，白紙の状態にもどしてから実行を開始するので，そのようなトラブルは起きない)．

File not found

「指定されたファイルがディスク上にない」

ディスク(ディスケット)の入れまちがい，ということもあるが，ファイル名の綴りの誤りということもよくある．これを調べるには

FILES　ドライブ番号 ↵

により，そのディスクに書き込まれているファイル名の一覧表を表示させてみるとよい．大文字と小文字は区別されるので注意．英字のŌと数字の0の混同ということもある．6字以上のファイル名を入れると，入れたつもりのないピリオドが挿入され，ピリオドを含めて指定しないと読めない．

File not open

「オープンしていないファイルを読もう（書こう）とした」

ŌPEN文の書き忘れ，または書きちがい．

Format over

「PRINT　USING文の書式指定が長すぎる」

For without NEXT

「FŌR文に対するNEXT文が無い」

Illegal direct

「その命令は直接実行形式では実行できない」

本機は大部分の命令が直接実行可能なので，めったにこのメッセージは出ない(たとえばDIM文やDEFFN文も直接実行できる．LABEL文やGŌSUB文をやっても，このメッセージは出ない)．

Illegal function call

「関数の使い方が悪い」

これは，たとえば平方根を計算する関数SQRの引数が負になった，というような場合である．ただちに引数の値を調べ，さらに，引数の値の計算の経過を調べて，引数がなぜ異常な値になったか，原因を究明する必要がある．たいていは平凡なミスであろうが，何度調べても

　　式はあっているのに値がおかしい

という場合には，丸め誤差が原因，ということが考えられる．本来なら $x \geqq 0$ となるはずの x が，丸め誤差のために，ほんのわずかではあるが，0より小さくなる，ということが判明したら，SQRを使用する前に

```
IF X<0 THEN X=0
```

という文を挿入すればよい．

Input past end

「入力ファイルの終端を過ぎたので入力不可能」

INPUT#文がデーターを要求しているが，既にファイルの終端に来ていて，入力できるデーターが無いと，という意味．

Line buffer overflow

「1行の字数が多すぎてバッファーに入りきれない」

一応上記の意味であるが，本機の場合，たいていは入れきれなくてもメッセージは出ず，はみ出した分は置き去りにして処理を進める．

Missing operand

「演算子の書き忘れ」

NEXT without FOR

「FŌR文がないのにNEXT文がある」

たいていは，FŌR文の書き忘れ，またはNEXT文の重複入力や，画面編集時の消し忘れが原因である．

Out of Data

「データー不足」

READ文で要求しているデーターがDATA文に書いていない．原因は，たいてい，単純なケアレス・ミスである．

Out of memory

「メモリー容量不足で実行できない」

Out of tape

「カセット・テープを入れて下さい」

自動的にふたがあいて，このメッセージが出る．

Overflow

「演算結果が過大のため処理不能」

実数型および倍精度実数型は絶対値が(約)10^{38}以上になると処理ができなくなる．また整数型は－32767～＋32767の範囲を越えると処理ができなくなり，このメッセージが出る．

REPEAT without UNTIL

「REPEAT文に対応するUNTIL文が無い」

Reserved feature

「その命令はディスク BASIC でないと使えない」

原因は二通り考えられる．ディスクを使用するプログラムの実行時にこのメッセージが出たのであれば，操作ミス(ディスク BASIC のシステムを入れていなかった) である．一方，ディスクを使うつもりがないのに，このメッセージが出たのであれば，何気なく書いた文がじつはディスク BASIC の命令であった，というケースが考えられる．

RESUME without error

「エラーがないのにRESUME命令が出た」

メイン・プログラムの最後のENDを書き忘れ，エラー処理ルーチンに迷い込んだ場合などにこのメッセージが出る．

RETURN without GOSUB

「GŌSUBで呼ばれていないのにRETURN文がある」

BASICのサブルーチンは独立したプログラム単位になっていないので，GŌTŌの行番号の書きちがいとか，メインプログラムの最後にENDが無く，すぐ続けてサブルーチンを書いてあったとか，その他いろいろな原因でサブルーチンの中に迷い込んでしまうことがあり，そのために上記のメッセージが出ることが多い．また，サブルーチンを呼ぶのにGŌSUBを使わずにGŌTŌを使ってしまったという場合もある．

String too long

「文字列の字数が多すぎて処理できない」

許されるのは255字までである．なお，文字列型定数の字数が長すぎた場合はエラーにならず，先頭255字だけが有効になるようである．

Subscript out of range

「添字の値が配列の寸法をオーバーしている（または負である）」

原因はさまざまであるが，常識的に予想される原因のほか，

配列の宣言をするのを忘れていた

第1添字と第2添字を混同して用いていた

関数名の誤り　(例)　Y=LN(100)

などということがある．

なお，DIM A(20000)　というような場合にも，どういうわけな，このメッセージが出る．

Syntax error

「文法違反がある」

原因としては，次のようなことが考えられる．

綴りの誤り (例) IMPUT A

記号の使い方の誤り (例) FŌR I=1,N

予約語を変数名に用いた (例) A=FŌR

構文上の誤り (例) ŌPEN "A" FŌR INPUT AS #1

GŌTŌ *A

Y=X**2

Tape read error

「カセット・テープの読み込み時のエラー」

Too complex

「式が複雑すぎて実行できない」

本機の BASIC はなかなかよくできていて，カッコを10重に使ったぐらいではこのメッセージは出ない．このメッセージが出るのは，たとえば次のような全く解釈不能なことをやった場合である．

```
10 X=2
20 DEFFNC(X)=1+FNC(X)
30 Y=FNC(X)
```

```
RUN
Too complex in 30
Ok
```

Type mismatch

「数値型と文字列型を混同している」

このメッセージを直訳すれば「型の不一致」ということになるが，整数型，実数型，倍精度実数型の間の演算，比較，代入等に際しては自動的に型変換が行なわれるので，このメッセージが出て停止したら，文字列型変数を書くべき場所に数値型変数を書いたとか，その逆とか，そういうたぐいの誤りであると思ってよい．

Undefined label

「定義してないラベルまたは行番号が参照された」

あとで飛び先のプログラムを書くつもりでいたが忘れてしまった，ということがよくある．そうでなければ，ラベルの綴りの誤りが考えられる(たとえば，GŌTŌ "GRAPH" に対し，行先の方では "GRAF" と書いていたりする)．修正時の誤操作により，必要な行を消してしまった，ということも考えられる．

Undefined function

「定義されていない関数が使用された」

ここでいう「関数」とは，利用者が定義するFN××という形の関数のことであって，FNの付いていないもの（たとえば，

```
Y=ASIN(X)
Y=LN(X)
Y=SQRT(X)
```

などと書いてしまった場合）は，このメッセージに含まれない（上の例の内，最初の二つは配列とみなされるので，Xが10以下ならば値は0となり，10以上または負ならば Subscript out of range というエラー・メッセージが出る．またSQRTは「予約語に始まる変数名が使われた」という理由でSyntax errorになる)．なお，関数のつもりでなく，変数としてFNで始まる名前を使ってもこのメッセージが出ることがある．

(例)

```
PRINT FNA
Undefined function
```

UNTIL withont REPEAT

「このUNTIL文に対応するREPEAT文がない」

WEND without WHILE

「このWEND文に対するWHILE文がない」

WHILE without WEND

「このWHILE文に対応するWEND文がない．

A.5 トレース

誤りの原因がなかなか見付からない場合，実行した文の行番号を実行順に表示することができる．これは，いわばプログラムの実行の足跡を調べるわけでトレース（trace）と呼ばれている．

トレースを行なうには，トレースを開始したい位置に**TRON**という文をおき，トレースを終了したい位置に**TROFF**という文を置けばよい．

（例）　以下に示すのは最大公約数を計算するプログラムの一例である．

```
10 INPUT M,N
20 IF M>=N GOTO 40
30 SWAP M,N
40 R=M MOD N
50 IF R=0 GOTO 90
60 M=N
70 N=R
80 GOTO 40
90 PRINT "GCD=";N
```

これのトレースを行うため，キーボードから

```
 5 TRON ⏎
95 TROFF ⏎
```

を入れて**TRON**と**TROFF**を付加し，実行させると次のようになる．

[　　] で囲まれた数字が，実行した行の番号である．

```
run
[10]? 24,30
[20][30][40][50][60][70][80][40][50][90]
GCD= 6
[95]
Ok
run
[10]? 123,456
[20][30][40][50][60][70][80][40][50][60]
[70][80][40][50][60][70][80][40][50][60]
[70][80][40][50][60][70][80][40][50][90]
GCD= 3
[95]
Ok
```

付録B　ASCIIコート表

DEC	HEX	CHR	DEC	HEX	CHR	DEC	HEX	CHR
32	20		64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
34	22	"	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	'	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
58	3A	:	90	5A	Z	122	7A	z
59	3B	;	91	5B	[	123	7B	{
60	3C	<	92	5C	¥	124	7C	¦
61	3D	=	93	5D	]	125	7D	}
62	3E	>	94	5E	^	126	7E	~
63	3F	?	95	5F	_	127	7F	

DEC	HEX	CHR	DEC	HEX	CHR	DEC	HEX	CHR
128	80	▁	160	A0		192	C0	タ
129	81	▂	161	A1	。	193	C1	チ
130	82	▃	162	A2	「	194	C2	ツ
131	83	▄	163	A3	」	195	C3	テ
132	84	▅	164	A4	、	196	C4	ト
133	85	▆	165	A5	・	197	C5	ナ
134	86	▇	166	A6	ヲ	198	C6	ニ
135	87	█	167	A7	ァ	199	C7	ヌ
136	88	▏	168	A8	ィ	200	C8	ネ
137	89	▎	169	A9	ゥ	201	C9	ノ
138	8A	▍	170	AA	ェ	202	CA	ハ
139	8B	▌	171	AB	ォ	203	CB	ヒ
140	8C	▋	172	AC	ャ	204	CC	フ
141	8D	▊	173	AD	ュ	205	CD	ヘ
142	8E	▉	174	AE	ョ	206	CE	ホ
143	8F	┼	175	AF	ッ	207	CF	マ
144	90	┴	176	B0	ー	208	D0	ミ
145	91	┬	177	B1	ア	209	D1	ム
146	92	┤	178	B2	イ	210	D2	メ
147	93	├	179	B3	ウ	211	D3	モ
148	94	▔	180	B4	エ	212	D4	ヤ
149	95	─	181	B5	オ	213	D5	ユ
150	96	│	182	B6	カ	214	D6	ヨ
151	97	▕	183	B7	キ	215	D7	ラ
152	98	┌	184	B8	ク	216	D8	リ
153	99	┐	185	B9	ケ	217	D9	ル
154	9A	└	186	BA	コ	218	DA	レ
155	9B	┘	187	BB	サ	219	DB	ロ
156	9C	╭	188	BC	シ	220	DC	ワ
157	9D	╮	189	BD	ス	221	DD	ン
158	9E	╰	190	BE	セ	222	DE	゛
159	9F	╯	191	BF	ソ	223	DF	゜

(注) DEC はコードを10進法で表したもの

HEX はコードを16進法で表したもの

CHR は対応する文字

DEC	HEX	CHR
224	E0	●
225	E1	○
226	E2	♠
227	E3	♥
228	E4	♦
229	E5	♣
230	E6	◢
231	E7	◣
232	E8	╳
233	E9	▘
234	EA	▖
235	EB	▝
236	EC	▗
237	ED	▞
238	EE	▚
239	EF	□
240	F0	▒
241	F1	土
242	F2	金
243	F3	木
244	F4	水
245	F5	火
246	F6	月
247	F7	日
248	F8	時
249	F9	分
250	FA	秒
251	FB	年
252	FC	円
253	FD	人
254	FE	生
255	FF	〒

0～31のコードは制御コードである．その内，

0～7はデーター通信の制御コード

（例）　2は**STX**（本文の始まり）

3は**ETX**（本文の終り）

8～15は出力制御コード

（例）　9は**HT**（水平**TAB**）

10は**LF**（1行進める）

11はカーソルを左上隅へ

12は**CL**（画面クリア，改頁）

13は**CR**（[↵]相当）

14は**SŌ**（シフト・アウト*）

15は**SI**（シフト・イン*）

16～23は特殊機能*

24～31はその他である．

（例）　24は**CAN**（キャンセル）

27は**ESC**（エスケープ**）

28は→
29は←
30は↑
31は↓
｝カーソルの移動

＊　効果は接続するプリンターなどの機種によって異なる．拡大文字，縮小文字の指令などに用いられる．

＊＊　標準のコード表から抜け出すためのコード．特殊文字や特殊制御に用いられる．

付録C　X1独特の命令および関数

テレビとのスーパーインポーズおよびテレビ制御は本機独特の機能で，これに関し，

ASK	自動ON，OFF用タイマーの設定
CHANNEL	チャンネル指定
CRT	テレビとコンピューターの切換え（重複も可）
SCRŌLL	コンピューター画面のスクロール
TVPW	テレビの電源のON，OFF
VŌL	音量指定

などの命令がある．詳しくは付録Eで説明する．

カセット・テープの制御はMZ-80BやMZ-2000ではできるが一般には珍しい．前方，後方，各50箇までのプログラムやデーターの自動頭出しができるようになったのは大きな進歩である．この関係の命令としては下記の五つがあり，詳しくは付録Fで説明する．

APSS	自動頭出し
CMT	カセットの状態を調べる．その他．
CSTŌP	停　止
FAST	早送り
REW	巻戻し

本機には，グラフィックRAMをデーターやプログラムの記憶のために転用できるという面白い機能があり，そのための命令および関数

ŌPTIŌN　SCREEN	用途切換え
DEVI$	読み出し
DEVŌ$	書き込み

などが用意されている．これについては付録Gで説明する．

グラフィック関係の内,

SCREEN　マルチ・ページの指定
CANVAS　表示画面の選択
WINDŌW　自動スケーリング
PALET　色番号の変更
PŌLY　正多角形の表示

などの機能は, 他のいくつかの機種にもあるが, なかなか便利で利用効果の大きい命令である.

LINE　直線を引く
CIRCLE　円を描く
PAINT　ぬりつぶす

などは, 命令としてはそれほど珍しいものではないが, 本機はオプション機能が充実しており, なかなか使いよく, たとえば

LINE文で点線を描ける
一つのLINE文で三つ以上の点を折れ線で結べる
CIRCLE文の角度指定は度（degree）単位で使い易い
PAINT文でタイリングが可能

などの特長がある.

文字の画面表示関係はなかなか強力である. まず,

CREV　反転表示
CFLASH　点滅表示

などは,機能的にはそれほど珍しくないが,命令が独立して使い易くなっている. 利用者がフォントやグラフィック・パターンを作って表示する機能に関しては, 強力な命令

CGEN　キャラクター・ジェネレーターの切換え
DEF CHR$　キャラクター・ジェネレーターの書き込み
PATTERN　グラフィック・パターンの表示

や, 関数

CGPAT	キャラクター・ジェネレーターに入っている文字のドット・パターンをビット・パターン・データーとして取り出す

が用意されており，手間をかければ相当に面白いことができる仕掛けになっている．漢字の表示に関しては，関数

KANJI$	漢字のドット・パターンを与える

がある(ただし漢字ROMが必要)．文字の表示の優先度を指定する命令

PRW	文字と図形のどちらを優先するかを指定

をもっていることも本機の大きな特長である．

INIT文については7章で少し説明したが，特長ある命令である．

INIT文　周辺装置に関するいろいろな指定を標準状態にもどす機能をもつ．書き方とその意味は次のとおり．

1）INIT "CRT:"　　または単に　　INIT

これは次のプログラムと同等の効果をもつ．

COLOR 7,0	黒地に白
CGEN	標準のキャラジェネ使用
CFLASH	点滅しない
CREV	反転しない
CSIZE	文字の大きさは普通
PALET	パレットコードを色番号に一致させる．
WINDOW	画面全域，画面座標使用
CONSOLE	全域スクロール
SCREEN 0,0,0	ページ0で書き込み，表示
PRW	図形よりも文字を優先

2）INIT "CAS:"

カセット・テープを巻き戻し，1巻全部を消去する．

それから，**DEVICE**という命令がある．カセット・テープだけのシステムではあまり利用価値がないが，ディスクや外部メモリー（増設RAM）などを使用する際には便利なものである．

DEVICE文　装置名のデフォルト値(指定を省略したときに用いられる標準指定値）を変更する．書き方は，

DEVICE "装置名："

で，装置名としては，たとえば

CAS	カセット・テープ
MEM	グラフィックRAM
EMM n	外部RAM（$n = 0〜9$）

などが指定できる．

関数では，3章で説明した

FAC(n **)**	階乗
SUM(n **)**	nまでの整数の和
RAD(x **)**	度をラジアンに変換
FRAC(x **)**	小数部

などは普通のBASICにないものであるが，そのほか，次のような面白い関数がある．

BIN$**　これは数値（ただし整数値に限る）を2進法表現の文字列に変換してくれる関数である．数値を16進法表現の文字列になおす関数**HEX$は多くの機種で使用できるが，2進法表現になおしてくれる関数は珍しい．ベテランなら**HEX$**さえあれば十分であるが初心者には**BIN$**は有用だと思う．書き方は

BIN$(数値型変数名または数式)

で，数値は16ビットの整数型に変換された上で，0と1の文字列に変換され，その上位の0を省いたものが**BIN$**の値になる．

(使用例)　入力した数値Nを2進法表現になおして出力するプログラムを作ってみよう．最も簡単に書けば

```
INPUT N
PRINT BIN$(N)
```

でよいはずであるが，これだけだとゼロ・サプレス（上位の不要な0を消す）機能のため，

0を入れれば出力は0

1を入れれば出力は1

となって面白くない．そこで，頭に0を補うため，

0を16字並べて**BIN$(N)**の左に連結する

その結果の右側16桁を関数**RIGHT$**で取り出す

という細工をする（ゼロ・サプレスをもとにもどすための定石）．プログラムおよび実行例は次のようになる．

```
10 INPUT N
20 B$=BIN$(N)
30 C$=RIGHT$(STRING$(16,"0")+B$,16)
40 LPRINT N,C$
50 GOTO 10
```

```
 3        0000000000000011
 2        0000000000000010
 1        0000000000000001
 0        0000000000000000
-1        1111111111111111
-2        1111111111111110
-3        1111111111111101
```

OCT$　　これは8進法表現に変換する関数である．書き方は

　　$\bar{\text{O}}$**CT$**(数値型変数名または数式)

[使用例]

```
10 INPUT N
20 B$=OCT$(N)
30 C$=RIGHT$(STRING$(4,"0")+B$,4)
40 LPRINT N,C$
50 GOTO 10
```

[実行例]

```
 1        0001
 10       0012
 100      0144
-1        7777
-16       7760
```

MIRRŌR$　これは，たとえば

10100111 ⟶ 11100101

というように，ビットの列を逆順に並べ換えてくれる関数である．このような機能は，グラフィックで左右対称な図形を表示する場合に便利であり，高速フーリエ変換などの際にも必要になる．これは普通の機能だけを用いてプログラムを書くと割合に手間がかかり，処理時間もかかるものであって，これが組込み関数として使用できるようになったことは非常に喜ばしい．

ただし，使用法に関しては，いろいろと制約があるので，下記の諸点に注意する必要がある．

1）引数は文字列型でなければいけない．

2）逆順変換はバイト内でのみ行なわれる．

（例）　1110001101000001

逆順　逆順

0100011110000010

3）逆順変換を行なうと，引数の内容も逆順になってしまう．

（例）　A$の内容が　11100011　のとき

B$=MIRRŌR$(A$)

とすれば，B$の内容は当然11000111になるが，同時にA$の内容も11000111に変ってしまう．A$の内容を変えたくなければ

B$=A$

B$=MIRRŌR$(B$)

とすればよい（逆順のまた逆順になって，もとにもどってしまうのではないかと心配されるかもしれないが，そうはならず，上記のようにすれば確かにB$はA$の逆順になる）．

[MIRROR$の使用および実行例]　キーボードから文字列を入力し，そのビット・パターンを逆順に変換するプログラムを作ってみよう．

```
10 INPUT A$
20 REPEAT
30 LPRINT A$
40 N=LEN(A$)
50 LPRINT "A$ ";
60 FOR I=1 TO N
70    C$=MID$(A$,I,1)
80    D$=RIGHT$(STRING$(8,"0")+BIN$(ASC(C$)),8)
90    LPRINT D$;" ";
100 NEXT I
110 LPRINT
120 B$=MIRROR$(A$)
130 LPRINT "B$ ";
140 FOR I=1 TO N
150    C$=MID$(B$,I,1)
160    D$=RIGHT$(STRING$(8,"0")+BIN$(ASC(C$)),8)
170    LPRINT D$;" ";
180 NEXT I
190 LPRINT : LPRINT
200 INPUT A$
210 UNTIL A$="*"
```

```
ABC
A$ 01000001 01000010 01000011
B$ 10000010 01000010 11000010
```

```
ｱｲｳｴｵ
A$ 10110001 10110010 10110011 10110100 10110101
B$ 10001101 01001101 11001101 00101101 10101101
```

```
♠♥♦♣
A$ 11100010 11100011 11100100 11100101
B$ 01000111 11000111 00100111 10100111
```

バイトの順序も逆にしたければ，行140以降を次のように変更すればよい．

```
140 E$=""
150 FOR I=N TO 1 STEP -1
160     C$=MID$(B$,I,1)
170     D$=RIGHT$(STRING$(8,"0")+BIN$(ASC(C$)),8)
180     LPRINT D$;"  ";
190     E$=E$+D$
200 NEXT I
210 LPRINT : LPRINT
220 INPUT A$
230 UNTIL A$="*"
```

```
ABCDEF
A$ 01000001 01000010 01000011 01000100 01000101 01000110
B$ 01100010 10100010 00100010 11000010 01000010 10000010
```

HEXCHR$　　ビット・パターンをプログラムの中に書く方法としては

&B 2進法の文字列　（例）　&B0101 ⎫
&Ō 8進法の文字列　（例）　&Ō237 ⎬ 整数型になる
&H16進法の文字列　（例）　&HFC2A ⎭

CHR$（&H16進法の文字列）⎫
後述のMKI$，MKS$，MKD$を用いる ⎭ 文字列型になる

などがあるが，上の四つは16ビット，MKS$で32ビット，MKD$でも64ビットまでしか使えない．ところが，グラフィック関係ではもっと長いビット・パターンを書きたいことがよくある．そういう場合，HEXCHR$を用いるとよい．これは

HEXCHR$（16進法の文字列）

（例）　HEXCHR$（"91E3"）

と書く．16進法の文字列は（文字列型として表せる限り，すなわち254桁までなら）いくら長くてもよい．その文字列は（16進法の2桁が1バイトになるから）先頭から2字ずつが一組になって対応する文字に変換される．したがって，文字列の長さは半分になる．

[備考] 引数の文字列の中に空白があると，そこがバイトの区切りとみなされる．空白が最初からバイトの区切りにあれば単に無視される．

(例1)

```
10 P$=HEXCHR$("1 2 3")
20 Q$=HEXCHR$("010203")
30 IF P$=Q$ THEN LPRINT "ｵﾅｼﾞ" ELSE LPRINT "ﾁｶﾞｳ"
```

```
RUN
ｵﾅｼﾞ
```

(例2)

```
10 S$=""
20 FOR I=&HB1 TO &HBF
30     S$=S$+HEX$(I)+" "
40 NEXT I
50 T$=HEXCHR$(S$)
60 LPRINT S$
70 LPRINT T$
```

```
B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
ｱｲｳｴｵｶｷｸｹｺｻｼｽｾｿ
```

MAXFILES　ファイル入出力の関係では**MAXFILES**という命令が書き方としてちょっと珍しい．これは同時にオープンできるファイルの数（じつはそれが「使用できるファイル番号の最大値」でもある）を指定するもので，

MAXFILES　同時にオープンするファイルの数

(例)　**MAXFILES 2**

と書く（特に指定しなければ1となる．また，16以上は指定できない）．この種の指定ができること自体は，べつに目新しいことではないが，他機種ではたいていシステムの起動時に指定するようになっているため，最初に使う人がこれを小さく設定してしまうとあとの人が困る（そのため，システム起動からやりなおさなければならない）という難点があった．それを命令として独立させたのはたいへん良いことだと思う．

MKI$, MKS$, MKD$, CVI, CVS, CVD などの関数は，ディスク BASIC にはたいてい付いているが，本機ではディスク BASIC でなくても使用できるため，普通の文字列処理やビット操作に利用できて便利である．簡単に説明すると，

MKI$	整数型データー（16ビット）を2バイトの文字列型データーとみなして扱う（読み換える）
MKS$	実数型データー（32ビット）を5バイトの文字列型データーとみなして扱う（読み換える）
MKD$	倍精度実数型データー（64ビット）を8バイトの文字列型データーとみなして扱う（読み換える）
CVI	2バイトの文字列を整数型データーとみなして扱う
CVS	5バイトの文字列を実数型データーとみなして扱う
CVD	8バイトの文字列を倍精度実数型データーとみなす

というもので，数値型データーをカセット・テープやディスクに記録する場合，

- **MKI$，MKS$，MKD$** などで文字列型になおして記録
- 再生した文字列は**CVI，CVS，CVD**などで数値型にもどすと

するのが本来の使用法である．具体的には，次のようにする．

(例)　n 箇のデーター $a_1, a_2, \cdots, a_n$ をカセット・テープに記録するプログラムは，最も平凡に書くと次のようになる．

```
60 OPEN "O",#1,"ﾃﾞｰﾀｰ"
70    FOR I=1 TO N
80       PRINT #1,A(I)
90    NEXT I
100 CLOSE #1
110 BEEP
```

これは割合に時間がかかる（たとえば $n = 300$ のとき，約1分半かかる）．カセット・テープやディスクに記録する場合，**PRINT#** の終りに ; 印を付けて（上の例では，行80を

```
80 PRINT #1,A(I);
```

にする）データーの間の空白をなるべく少なくする方がよい，といわれており，同様なねらいで後述の**WRITE#**文を用いる，というのも一案

であるが，上記の例のような場合には，あまり効果がない．

上の方法でなぜ時間がかかるかといえば，ディスプレイやプリンターに出力するための文（PRINT#文は，本来，そういう命令である）を用いている関係上，データーはすべて10進法の文字列になおして記録されることになり，表には見えないけれども，ずいぶんムダなことをやっているわけである（その結果，記録するビット数が多くなってしまい，書き込み，読み出しに時間がかかる）．

これを避けるにはデーターを内部表現（2進法の形）のままカセット・テープに記録すればよい．そのために**MKI$，MKS$，MKD$，CVI，CVS，CVD**などの命令があるわけである．しかし，これを使うには，ちょっとコツがいる．平凡な考え方でいくと，

```
PRINT#1, MKS$(A(I))
```

で書き込んで

```
INPUT#1, B$
A(I)=CVS(B$)
```

とすればよさそうであるが，それでは正しく読めない．

INPUT#文はINPUT文の仲間で，もともとはキーボード入力のための命令であり，「字数を指定しないで入力できる」という点に大きな特長がある．たとえばFORTRANの場合*には入力データーの字数（桁数）をプログラムで指定する方式になっており，キーボードから入力する場合でも，プログラムで指定された字数に合わせて（データーの方が短かすぎたら空白を付けて）入力しなければならない．それに対し，BASICのINPUT文は，データーの字数（桁数）を指定する必要がなく，入力の字数が自由なので非常に使い易い．そういうことができるのは，データーの区切りを区切り記号（INPUT文の場合，コンマまたは[↵]）で検出しているからである．

---

* FORTRAN 77やそれに類似の拡張FORTRANではBASICと同様の扱いも可能であるが．

普通の文字列の入力にはそれでよい．しかし内部表現(2進法)のビット列を8ビットずつ区切って文字とみなして扱う(**MKS$**等の)場合には少々ぐあいの悪いことがある．まず考えられることは，ビットのパターンが偶然，

　　コンマを表すコード　00101100

　　⏎を表すコード　　00010011

に一致したら困る，ということであろう．このようなパターンが入ってくると，それを区切り記号とみなしてしまう．文字列データーとしては

　　"を表すコード　　00100010

もトラブルの原因となる．もっと困るのは

　　ヌル・コード　　00000000

で，これはBASICでは文字として扱われない仕組みになっている．たとえば，カセット・テープにこれが記録されていても「これは文字でない」という理由で無視されてしまう(入力されない)．

そういうわけで，**INPUT#**文は**MKS$**等で変換されたデーターの入力に用いることはできない．それではどうすればよいかというと，要するに「字数(桁数)指定方式の入力命令」で読めばよい．

BASICで使用できる「字数(桁数)指定方式の入力方法」としては**INPUT$**という関数がある．これはファイル入力にも使用することができて，その場合は次のように書く．

```
       INPUT$(字数,#ファイル番号)
(例)   INPUT$(5,#1)
```

これを用いて作ったプログラムの一例を以下に示す．行8000から8040まではテスト・データーを生成するプログラムで，書き込みのプログラムは行8060から8140までである．後日，これを再生するには行8500～8600のプログラムを実行すればよい．

```
8000 INPUT N
8010 DIM A(N)
8020 FOR I=1 TO N
8030     A(I)=1/I
8040 NEXT I
8050 LABEL "ｶｷｺﾐ"
8060 INPUT "ﾌｧｲﾙﾒｲﾊ?",F$
8070 OPEN "O",#1,F$
8080     FOR I=1 TO N
8090         PRINT #1,MKS$(A(I));
8100     NEXT I
8110 PRINT #1
8120 CLOSE #1
8130 APSS -1
8140 END
8500 LABEL "ﾖﾐｺﾐ"
8510 INPUT N
8520 DIM A(N)
8530 INPUT "ﾌｧｲﾙﾒｲﾊ?",F$
8540 OPEN "I",#1,F$
8550     FOR I=1 TO N
8560         B$=INPUT$(5,#1)
8570         A(I)=CVS(B$)
8580         PRINT A(I)
8590     NEXT I
8600 CLOSE #1
```

［注意］ これで一応は動くけれども，このプログラムには一つ，大きな欠点がある．それは，ビット・パターンとして

　　BREAK のコード　　00000011

が入ってくるとデーターとはみなされず，BREAK キーが押された，と解釈されてしまう（**INPUT$** の文法がそのように定められている）からである．あいにく本機では **ŌN STŌP GŌSUB** という文を使えないので，この難点は簡単には解決できない．

それでは，どうすればよいかというと，**DEVI$**，**DEVŌ$** という命令で入出力を行なえばよいのである*．これは1レコード（256バイト）単位で入出力を行なう命令で，他機種のディスク BASIC のランダム・アクセス・ファイル用の命令 **GET**，**PUT** に相当するものであるが，使い方は少し異なり，次のように書く．

---

* **DEV** は device，**I** は input，**Ō** は output の意味．

(入力)

DEVI$　"装置名：",レコード番号,変数名1,変数名2

(出力)

DEVŌ$　"装置名：",レコード番号,変数名1,変数名2

(例)　**DEVI$　"CAS:",1,P$,Q$**

　　　DEVŌ$　"CAS:",1,F1$,F2$

ここで「変数名1」,「変数名2」と書いてあるのは,

(入力時)　入力したデーターの格納先

(出力時)　出力すべきデーターが入っているところ

を表す文字列変数名で,なぜ二つに分かれているかというと,入出力単位(1レコードの長さ)は256バイトであるが,文字列型データーは最大255バイトまでしか扱うことができないため,前半と後半に分け,各128バイトずつ入れるようにしているのである.出力時(命令**DEVŌ$**)の変数名1,変数名2の内容は,それぞれが,ちょうど128バイトでなければいけない(違反すると文法エラーということで止まってしまう).

[備考]　装置名としては,カセット・テープを表す**CAS**のほか

　　MEM　　　グラフィック・メモリー

　　EMM *n*　　外部メモリー($n = 0〜9$)

などを書くことができる.

DEVI$**,**DEVŌ$を使うためには,データーを256バイトずつまとめて(あるいはもう少し小さな単位でまとめて後に詰め物を入れて256バイトにして)記録し,再生後それをまた仕分けしなければならないから手間がかかる.参考までにそのようなプログラムの一例を以下に示す.このようにすれば記録,再生の所要時間を,最初に示した「最も簡単な方法」の1/3以下に短縮することができる.

```
8000 REM ▒▒▒▒▒▒▒▒ MKS SUB ▒▒▒▒▒▒▒▒
8010 REM ｱﾗｶｼﾞﾒ ﾃｲｷﾞｼﾃｵｸﾍﾞｷ ｱﾀｲﾊ
8020 REM     N ｵﾖﾋﾞ A(1)･･･A(N)
8030 LABEL "ｶｷｺﾐ"
8040 DIM B$(2)
8050 M=1
8060 MADE=N¥50+1
8070 FOR K=1 TO MADE
8080   FOR J=1 TO 2
8090    B$(J)=""
8100     FOR I=1 TO 25
8110       B$(J)=B$(J)+MKS$(A(M))
8120       IF M=N GOTO "fill"
8130       M=M+1
8140     NEXT I
8150   NEXT J
8160   LABEL "fill"
8170   FOR J=1 TO 2
8180     L=LEN(B$(J))
8190     B$(J)=B$(J)+STRING$(128-L,"U")
8200   NEXT J
8210   DEVO$ "CAS:",K,B$(1),B$(2)
8220   NAPSS=K
8230   IF M=N GOTO "ｵﾜﾘ"
8240 NEXT K
8250 LABEL "ｵﾜﾘ"
8260 APSS -NAPSS
8270 END
8500 LABEL "ﾖﾐｺﾐ"
8510 INPUT "N=",N
8520 DIM A(N),B$(2)
8530 M=1
8540 MADE=N¥50+1
8550 FOR K=1 TO MADE
8560   IF M>=N GOTO "END"
8570   DEVI$ "CAS:",K,B$(1),B$(2)
8580   FOR J=1 TO 2
8590     FOR I=0 TO 24
8600       C$=MID$(B$(J),5*I+1,5)
8610       A(M)=CVS(C$)
8620       IF M=N GOTO "END"
8630       M=M+1
8640     NEXT I
8650   NEXT J
8660 NEXT K
8670 LABEL "END"
8680 FOR I=1 TO N
8690   PRINT A(I)
8700   IF (I MOD 10)=0 THEN PAUSE 30
8710 NEXT I
```

以上のほか，ちょっと珍しい関数としては次のようなものがある．

DTL　これは，いわば「DATA文の現在位置」を教えてくれる関数で，正確にいえば「次に読み込まれるDATA文の行番号」がDTLの値になる．ただし最後のデーターを読み終ったあとは，最後のDATA文の行番号になる．引数は不要で単にDTLと書く．

（例）

```
10 DATA 1
20 DATA 2
30 DATA 3
40 DATA 4
50 DATA 5
60 FOR I=1 TO 5
70    READ A(I)
80    LPRINT DTL,A(I)
90 NEXT I
```

（実行結果）

```
20    1
30    2
40    3
50    4
50    5
```

KEY 0　これはキーボード入力バッファーに文字列を書き込む命令であって，次のように書く．

KEY　0，"文字列"

この文を実行すると，" "内に書かれた文字列が，そのときキーボードから入力されたのと同じことになる．

［注意］　文字列の長さは63字以下でなければいけない．64字以上だと後の部分は無視される．

（応用例）　DATA文やPLAY文をたくさん書くとき，いちいちDATAとかPLAYという綴りを入れるのはめんどうである．そこで，行番号を自動的に発生し，さらにDATAという文字も付けてくれるプログラムを作ってみた．あとは正味のデーターだけをキーボードから入れればよい．入力されたプログラムは，一応「データー」として文字列型配列A$に格納される．それをプログラムとして利用するには，A$の内容をファイル（カセット・テープまたはグラフィック・メモリー）に書き込んで，あらためてそれをMERGEすればよい．

```
10 INPUT "ｺｽｳ";N
20 DIM A$(N)
30 FOR I=1 TO N
40   KEY 0,STR$(I*10)+" DATA "
50   INPUT A$(I)
60 NEXT I
70 REM ﾌﾟﾘﾝﾄｼﾃﾐﾏｼｮｳ
80 FOR I=1 TO N
90    LPRINT A$(I)
100 NEXT I
```

(実行例)　行10では7を入れ，行50で

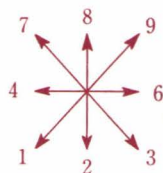
SUN　MŌN　TUE　WED　THU　FRI　SAT

を入れれば，出力（すなわち配列**A$**の内容）は次のようになる．

```
10 DATA SUN
20 DATA MON
30 DATA TUE
40 DATA WED
50 DATA THU
60 DATA FRI
70 DATA SAT
```

次の六つは他の二，三の機種にも同様な機能をもつものがあるが，一応ここで説明しておく．

REPEAT OFF **REPEAT ON**	キーボードのオート・リピート機能の停止（OFF）および再開（ON）
CLICK OFF **CLICK ON**	キーボードのクリック音の停止（OFF）および再開（ON）
STICK（n）	ジョイスティックの向きを知る
STRIG（n）	ジョイスティックのボタンの状態を知る

ただしnはジョイスティックの番号（1または2）で，$n=0$とすれば「テン・キーで代用」ということになる（押しボタンはスペース・バーで代用）．向きの表し方は右図のとおりで要するに「テン・キーの配列と同じ」．関数**STRIG**の値は押されているとき−1，押されていないとき0となる．

最後になったが，SEARCHというコマンドについて説明しておく．これはHuBASIC独特のコマンドで，MZ-80C/Kの頃から非常に重宝がられたものである．

機能は，現在メモリーにロードされているプログラムを行番号順に調べていって，指定した綴りが文の中にあったらその行番号を表示する．指令方法は次のとおり．

```
SEARCH "綴り"
```

用途としては，たとえば次のようなものが考えられる．

1）ある変数名がどこに使われているかを調べる．

2）あるラベルがどこに書いてあるか，またどこで参照されているかを調べる．

3）ある命令の使用箇所を調べる．たとえばDIMを全部リスト・アップすれば，使用されている配列が全部わかる．

4）変数の値がどこで書き換えられているかを調べる．たとえば，変数Aについて調べるのであれば，

```
SEARCH "A="
```

とすればよい（あと，入力関係の命令をリスト・アップしてみる）．

5）ある定数がどこに書いてあるかを調べる．デバッグの際にうまく使うと役に立つ．

付録D　割り込み処理機能の解説

D.1　割り込みとは

割り込み（interrupt）とは，一つのプログラムの実行中に，緊急のプログラムを優先的に実行するため，今の仕事を中断させて緊急の仕事に切り換えることをいう．

ただし，単に BREAK キーで実行を中断し，あとで実行を再開するための情報（行番号など）をメモし，別のプログラムをロードして実行させる，…というようなのは，普通，「割り込み」とは言わない．狭い意味では，これら（今のプログラムの中断，緊急のプログラムへの切り換え，以前のプログラムの実行再開）が全部自動に行なわれる場合を「割り込み」と呼んでいる．

具体的には，本機の場合

　　ファンクション・キーを押したとき

　　実行中に異常事態が発生したとき

などに，自動的に

　　実行中のプログラムを中断し，

　　処理プログラムを起動し，

　　処理終了後，以前のプログラムを再開する

ことができる．

D.2　プログラミングの要領

割り込み処理を行なうには，次のことが必要である．

1）処理プログラムの作成とロード

割り込んで実行すべきプログラムを作成し，メモリーに入れておく．

2）入口の行番号の登録

どういう種類の割り込みがあったときに，どのプログラムを実行させるか，ということを

ŌN　割り込みの種類　GŌSUB　入口の行番号

3）割り込み許可，禁止，中断

あまり勝手なときに割り込みをされては困ることが多いので，ファンクション・キーの割り込みに関しては割り込みを許可したり禁止したり中断（保留）したりできるようになっている．それには

KEY　番号　ŌN　……許可
KEY　番号　ŌFF　……禁止
KEY　番号　STŌP　……中断

という文を用いる．この内，中断というのは，「割り込み要求があった」という情報だけを記憶して，本来のプログラムの実行を続け，割り込み許可の下り次第，割り込み処理プログラムに切り換える，という方式である．

BASICの割り込み処理は，このように，メイン・プログラムと割り込み処理プログラムが一体になっている（独立していない）．いわば「自分のことは自分でせよ」という方式である．こういう方式には不便な面もあるが，単純明快であり融通がきく．

D.3　ファンクション・キーによる割り込み

キーボードの上部に並んでいる[F1]～[F5]のキーをファンクション・キーという．これは普通，2章で説明したように簡便操作の目的に用いられているが，本来は割り込み用のスイッチで，これらのキーを押すことにより，あらかじめ指定しておいた行番号にジャンプし，割り込み処理経了後，以前のプログラムを続行させることができる．それには

　　ŌN　KEY　GŌSUB　F1の行先，F2の行先，…

により，F何番を押したらどこに飛ぶかを指定し，

　　KEY　キー番号　ŌN

により割り込み許可を与えればよい．なお，割り込みの必要がなくなったら，ただちに

　　KEY　キー番号　ŌFF

により割り込み指定を解除しておかないといけない（そうしないと簡便操作の目的に使えない）．

```
1 INIT
2 CLS 4
10 ON KEY GOSUB 8000
20 KEY 1 ON
3000 REM --- ﾕｰｻﾞｰ ﾉ ﾌﾟﾛｸﾞﾗﾑ ﾉ ﾚｲ ---
3010 FOR J=0 TO 195 STEP 5
3020 FOR I=1 TO 199 STEP 3
3030    LINE (J,J)-(I,I),PSET,(I MOD 8),B
3040 NEXT I
3050 NEXT J
3060 END
8000 REM --- F1ﾜﾘｺﾐｼｮﾘ ---
8010 INIT
8020 CLS 4
8030 SCREEN 1,1
8040 CLS 4
8050 SCREEN 0,0,0
8060 KEY 1 ON
8070 RETURN
```

D.4　異常事態による割り込み

異常事態（エラー，たとえば0による割り算，メモリー不足，文法違反等）が起こると，普通はエラー・メッセージを表示して停止するが，あらかじめ

　　ŌN　ERRŌR　GŌTŌ　エラー処理の開始番号またはラベル

という文を実行しておくと，この文で指定された行に飛んで，エラー処理を行なうことができる．エラー処理後は，

　　RESUME　NEXT

と書けば異常事態発生箇所の次の文から実行を再開することができ，

　　RESUME　行番号

と書けば，そこで指定した行番号から実行を再開することができる（普通のGŌTŌ文で帰ると，「エラー処理が終った」という報告をしていないことになり，そのため，再度エラーが起ったときに割り込みせずに停止してしまう）．

エラー処理プログラムの中でエラーの発生箇所やエラーの種類を知ることができるように**ERR**および**ERL**という特別な変数が用意されていて，

　　ERRにはエラーの種類を表す番号（エラー・コード）

　　ERLにはエラーを起こした文の行番号

が入る．エラー・コードは次表のとおり（各メッセージの詳しい説明は付録A.4参照）．

D.5　エラー・コード表

1 NEXT without FOR
2 syntax error
3 RETURN without GOSUB
4 out of data
5 illegal function call
6 overflow
7 out of memory
8 undefined label
9 subscript out of range
10 duplicate definiton
11 division by zero
12 illegal direct
13 type mismatch
14
15 string too long
16 too complex
17 can't continue
18 undefined user function
19 no RESUME
20 RESUME without error
21 illegal format
22 missing operand
23 line buffer overflow
24
25 bad screen mode
26 UNTIL without REPEAT
27 out of tape
28
29 tape read error
30 bad file mode
31 out of stack
32 WHILE without WEND
33 WEND without WHLE
34 reserved feature
35 FOR withont NEXT
36 format over
37 REPEAT without UNTIL
50 FIELD overflow
51 device in use
52 bad file number
53 file not found
54 already open
55
56 device I/O error
57 file already exists
58
59
60 device full
61 input past end
62
63
65 bad allocation table
66 bad file descripter
66 bad record
67 no password
68
69
70
71 file not open
72 write protected
73 device offline

付録E　テレビの制御とスーパーインポーズ

E.1　概　　説

本機のテレビ制御は

- 内蔵の予約タイマー・システムで制御する
- BASICのプログラムで制御する

の二本立てになっている.

前者で制御できるのは

　　指定された時刻に電源を入れる（または切る）
　　指定されたチャンネルに切り換える

の二つで，これを

　　毎日行なう
　　毎週，指定された曜日に行なう
　　特定の指定された日にだけ行なう

ことなどが可能である．テレビを見るだけなら，これだけできれば十分であろう．

一方，BASICでできることは，

　　コンピューターの出力をテレビ映像に重ねて表示する

ことが中心で，これに関連して

　　電源のON，OFF　　チャンネルの切換え
　　音量の調節　　コンピューターで表示した内容のスクロール

などが可能である.

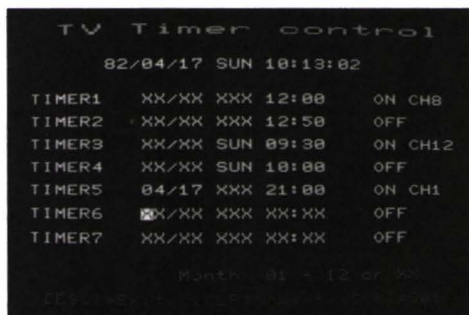
[注意]　予約タイマー使用期間中は，主電源スイッチ（本体の後のスイッチおよびディスプレイのふたの中のスイッチ）を切ってはいけない（もちろんコンセントを抜いてもいけない）．停電があったあとは，予約内容を再設定する必要がある.

E. 2　予約タイマー・システムの使用法

これはBASICと独立に(BASICを知らなくても，BASICをロードしてなくても) 使用できるようになっており，なかなか使い易くできている．使用手順は次のとおり．

本体の電源を入れると，

```
Make ready any device
Push (F,R,C or T) key
      F:FLoppy
      R:ROM
      C:CMT
      T:Timer
```

という表示が出る．普通はここでBASICのシステム・テープを読み込ませるわけであるが，予約タイマー設定のときは**Timer**を指定する．それにはTのキーを押せばよい．そうすると次のような画面になる．

- 1行目はタイトル
- 2行目は今日の日付と曜日と現在時刻
- 3～9行目は七つのタイマーの設定内容
- 点滅しているのはカーソルで，上下左右自由に動かしてよい
- 10行目は現在のカーソル位置で入力できるデーターの種類と，許される値の範囲
- 11行目は特殊キーの説明

現在の日付と曜日と時刻の設定方法　本機を購入して最初に使うときは日付が狂っている．主電源を切ってあっても日付が狂う．正しい日付，曜日，時刻を設定するには，カーソルを2行目にもっていって，

年／月／日　曜日　時：分：秒

をキーボードから入れて，[↵]キーを押せばよい．データーの表し方は画面の下から2行目に黄色で表示されている．たとえば，カーソルを曜日の所にもっていくと，下から2行目に

SUN MŌN TUE WED THU FRI SAT or XXX

と表示される．この中から今日の曜日を選んで入力する．

[注意]　年，月，日，時，分，秒は必ず2桁で入れること．たとえば1月なら**01**，また，0分ならば**00**と入れる．

予約タイマーの設定方法　予約タイマーは七つあって，その設定内容が画面に表示されている．最初は

XX／XX　XXX　XX：XX　　ŌFF

となっている．各欄の意味は

月　日　曜日　時　分　入切の別

で，XX印は「指定しない」(指定されていない)ということを表す．カーソルは最初，左端（月の位置）にある．何月何日ということを指定したければ，ここで月と日を入れる（1桁の場合は頭に0を補って2桁にして入れること)．指定する必要がなければ，カーソル移動のキー[→]を押すと次の項目に移る（次の桁に移るのではない．したがって，たとえば**02**を**03**に変更したいときに「**0**は共通だから[→]キーで1字進めて**3**だけ入れよう」などということはできない．**ŌN**，**ŌFF**の指定の欄に来ると，下から2行目に黄色い文字で

TV POWER ŌN?　(Y or N)

という指示が出るから

ŌN　　にしたければ　Y

ŌFF　にしたければ　N

のキーを押す．ここで**Y**を押すと，文字**ŌN**の右に**CH**と表示されるから，チャンネル番号を入れる（ここは1桁でもよい)．

1行分,出来上ったら [⏎] キーを押すと,内蔵タイマーに登録され,「登録された」という印に赤い＊印が表示される．このあたりの要領は，BASICのプログラムを入力するときのスクリーン・エディターと同様で，カーソルを「修正したい箇所」にもっていって新しい値をキーボードから入れ，[⏎] キーを押せばよい．

(予約指定の例)

- 4月5日12時55分にスイッチを入れ1チャンネルにする
  `04/05 XXX 12:55 ŌN CH1`
- 毎週日曜日の13時35分から14時25分まで1チャンネルを見る
  `XX/XX SUN 13:35 ŌN CH1`
  `XX/XX SUN 14:25 ŌFF`
- 毎日，12時から12時50分まで8チャンネルを見る
  `XX/XX XXX 12:00 ŌN CH8`
  `XX/XX XXX 12:50 ŌFF`

予約の取り消し　解除したい場合はカーソルを該当する行に合わせて [SHIFT] キーを押しながら [CLR/HŌME] キーを押す．

BASICにもどる方法　予約タイマーの設定が終ったら [ESC] (エスケープ) キーを押すと，もとの画面

```
Make ready any device
Push (F,R,C or T) key
```

にもどるから，カセット・テープを入れてCを押せば，以後，普通のパソコンとして使える．「テレビŌN」の時刻になると，画面が自動的にテレビの方に切り換わる．

BASICの途中で予約タイマーの設定を行なう方法　このためにASKというコマンドがある．使用法は，(WIDTH 80で使っている場合はまずWIDTH 40 [⏎] としてから)

```
ASK ⏎
```

とすればよく，これで TV Timer control の画面になる．設定を終って [ESC] キーを押すと，BASICのコマンドを受け付けられるモードにもどり，Ōk が表示される．

E.3 BASICで制御する方法

電源のON, OFF

TVPW ŌN　　ディスプレイ装置の電源を入れる

TVPW ŌFF　　ディスプレイ装置の電源を切る

テレビとコンピューター出力の切換え

CRT 0　　テレビ映像を表示

CRT 1　　コンピューター出力を表示

CRT 2　　重ねて表示(テレビのコントラストを下げる)

CRT 3　　重ねて表示(同じコントラストで表示)

チャンネルの指定

CHANNEL チャンネル番号

音量を変える(瞬時には変らないでちょっと時間がかかる)

VŌL 変化量

変化量としては−62以上, 63以下の値を指定でき, 正なら音が大きくなり, 負なら音は小さくなる.

コンピューター出力をスクロール表示(画面の外に出た行は, 反対側の端からまた入ってきて, ぐるぐるまわる)

SCRŌLL 3　　上へ速くスクロール

SCRŌLL 2　　上へスクロール

SCRŌLL 1　　上へゆっくりスクロール

SCRŌLL 0　　スクロール停止*

SCRŌLL −1　　下へゆっくりスクロール

SCRŌLL −2　　下へスクロール

SCRŌLL −3　　下へ速くスクロール

ただしテレビとコンピューターの重複表示のときのみ有効

---

* スクロール中にキーボードからこれを入力することはできないので, 手動で停止させたいときは SHIFT + BREAK を用いる.

[簡単な使用例]

1）**チャンネル・レビュー**　テレビのチャンネルを

1，　3，　4，　6，　8，　10，　12

の順に切り換えて2秒間ずつ表示するプログラムを作ってみよう．

```
10000 REM --- ﾁｬﾝﾈﾙ ﾏﾜｼ ---
10010 LABEL "CCC"
10020 CRT 2
10030 DATA 1,3,4,6,8,10,12
10040 RESTORE
10050 FOR I=1 TO 7
10060     READ C
10070     CHANNEL C
10080     PAUSE 20
10090 NEXT I
10100 CRT 1
10110 RETURN
```

行番号を10000からにしたのは，これを常時入れておいて，気が向いたときに

```
GŌSUB 10000
```

としてチャンネルをまわしてみるためである．ひんぱんに使うのなら

```
KEY 3,"GŌSUB10000"+CHR$(13)
```

というぐあいにファンクション・キーに登録しておくとよい（こうしておけば F3 キーを押すだけで全チャンネルが見られる．

2）**音量テスト**　音量を，だんだん大きくしていって最大にして，それからだんだんと小さくしていって最小にするプログラムの例を示す．

```
10 REM --- volume control ---
20 REM ｲﾁﾊﾞﾝ ﾁｲｻｸｽﾙ
30 VOL -62
40 PAUSE 30
50 CRT 2
60 REM ｵｵｷｸｽﾙ
70 FOR I=1 TO 62
80    VOL 1
90    GOSUB 180
100 NEXT I
110 REM ﾁｲｻｸｽﾙ
120 FOR I=62 TO 1 STEP -1
130    VOL -1
140    GOSUB 180
150 NEXT I
160 CRT 1
170 END
180 REM ﾚﾍﾞﾙ ｦ ﾋｮｳｼﾞ
190    CLS
200    CSIZE 3
210    COLOR 2
220    PRINT I
230    PAUSE 1
240 RETURN
```

付録F　カセット・テープの制御

本機では，テープ・レコーダーの操作をすべて自動に（プログラム制御で）行なうことができる．

機　能	説　　明	書　き　方
EJECT	カセット・テープ装置のふたをあける	EJECT
STŌP	走行停止	CSTŌP
READ	録　音*	CMT=2*
FF	早送り	FAST
REW	巻戻し	REW
APSS	前方 *n* 箇（－を付ければ後方 *n* 箇）のファイル** をスキップして自動頭出しを行なう	APSS　*n*
WRITE	再　生*	CMT=10*

（例）　APSS　3　　　プログラム3本を読みとばす

　　　APSS　－1　　いま入力（または出力）したファイルの先頭まで巻き戻す．

---

*　コンピューターへの読み込みやコンピューターからの書き込みはINPUT#文，PRINT#文で行なうが，普通のテープ・レコーダーとして使う場合には，このCMT文によって指令する．

**　プログラムは1本が一つのファイルになる．データーはŌPENしてからCLŌSEするまでが一つのファイルになる．

[APSSの使用例]

```
10 OPEN "O",#1,"DATA1"
20     FOR K=1 TO 5
30         PRINT #1,K
40     NEXT K
50 CLOSE #1
60 OPEN "O",#1,"DATA2"
70     FOR K=1 TO 5
80         PRINT #1,K+5
90     NEXT K
100 CLOSE #1
110 APSS -1
120 OPEN "I",#1,"DATA2"
130     INPUT #1,A,B,C,D,E
140 CLOSE #1
150 PRINT A;B;C;D;E
```

[実行結果]

```
 6  7  8  9  10
```

カセット・テープの状態を調べる方法　CMTという関数を用いて次のような情報を得ることができる．

書き方	関数の値	意　味
CMT(0)	−1	テープが動いている
	0	テープが止まっている
CMT(1)	−1	テープが入っている
	0	テープが入っていない
CMT(2)	−1	書き込み禁止のつめを折っていない
	0	書き込み禁止のつめを折ってある

[解説]　先に4.19節で説明したように，論理変数として使うと

−1は真（条件成立）　　0は偽（条件不成立）

を表すので，たとえば

IF CMT(1) THEN … ELSE …

というような使用法もできる．

付録G　グラフィックRAMをメモリーに転用する方法

本機には48Kバイトのグラフィック RAM が付いている（オプションということになっているが，本機をグラフィックなしで使う人は少ないであろう）．48Kバイトいえば，かなりの容量である．単精度実数型のデーターなら9600箇も収容できるし，プログラムなら（1行の平均字数にもよるが）2000行ぐらいは入るはずである．グラフィック表示の必要のないとき，これを普通のメモリーとしてデーターやプログラムの記憶に使うことができれば便利であろう．本機の場合メイン・メモリーのユーザー領域が約23Kバイトであるから，これに48Kバイトもプラスできたら約3倍の広さになるわけである．他機種では，たいてい，そんなことはできない．「グラフィック RAM を転用できたらいいねえ」と私達はよく話し合っていたが，夢であった．その夢が本機でようやく実現された．非常に喜ばしいことである．

ただし，普通のメモリーと全く同じに使用できるわけではない．高速の外部記憶装置として使うのである．MZ-80B/C/K 用に I/O データー機器から「大容量 RAM」というのが発売されているが，あれと同じような感じで使うのである．使用法は，詳しくは次ページで説明するが，カセット・テープやディスクとだいたい同じで

SAVE文でプログラムをセーブする

LŌAD文でプログラムをロードする

PRINT#文でデーターを書き込む

INPUT#文でデーターを読み込む

などが可能である．ディスクと比較すると容量は小さいが，一時的な記憶場所として使うのであれば，この程度でもかなり役に立つ．また，カセット・テープよりも断然速くて便利である．

転用宣言　グラフィックRAMをメモリーに転用するには

```
OPTION SCREEN 2
```

と宣言（命令）すればよい．逆に，再びグラフィックRAMとして使用する場合には

```
OPTION SCREEN 1
```

とする．

初期化　次に，ファイル管理プログラムの初期化（状態を白紙にもどすこと）を行なう必要がある．それには

```
INIT "MEM:"
```

と命令すればよい．

プログラムのロード，セーブ

プログラムのロードは　LOAD "MEM:ファイル名"

プログラムのセーブは　SAVE "MEM:ファイル名"

データーの入出力　データーの入出力を行なうには，まず

入力は　OPEN "I",#番号,"MEM:ファイル名"

出力は　OPEN "O",#番号,"MEM:ファイル名"

でオープンしておいて，**PRIN#**文，**INPUT#**文で記録，再生を行ない，最後に**CLOSE**文でファイルを閉じればよい．

WRITE#　**LPRINT#**　**INPUT$**

DEVI$**　**DEVO$

FILES　**LFILES**

なども使用できる．

付録H　乱数RNDとその応用

関数RNDは0から1までの間の一様分布の乱数を発生する．引数は書かないでよく，単にRNDと書けばよい．

実際にどんな数が出てくるか実行して調べてみよう．

```
10 FOR I=1 TO 10
20     U=RND
30     LPRINT U
40 NEXT I
50 END
```

［実行例］

```
.77344871
.3838042
.76547372
.64453149
.96692097
.24500155
.82409966
.43847716
4.6308994E-02
.21497321
```

用途によっては，［0，1］以外の区間の一様乱数が必要となることがある．それには変数変換すればよい．たとえば，0から99までの整数の乱数を作りたければ，

INT(RND*100)

とすればよい．

```
10 FOR I=1 TO 10
20     U=INT(RND*100)
30     LPRINT U;
40 NEXT I
50 LPRINT
```

［実行例］

```
50  55  26  1  84  19  44  4  75  78
```

また，6倍して整数部をとれば，サイコロの代用品として使える．

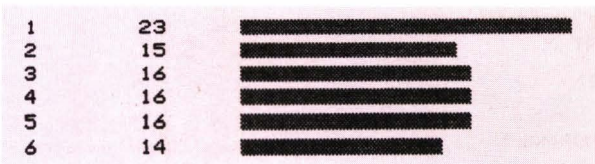
```
10 FOR I=1 TO 10
20     U=INT(RND*6)+1
30     LPRINT U;
40 NEXT I
50 LPRINT
```

［実行例］

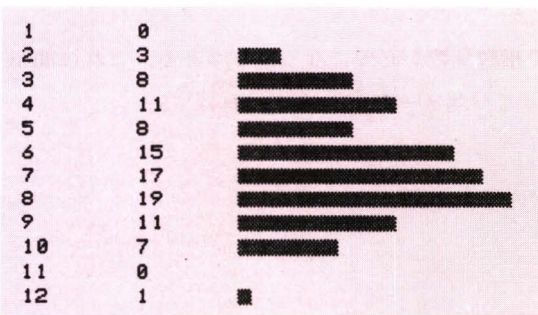
```
5  3  3  6  3  1  3  4  2  6
```

こうして作られる乱数は本当に「一様」になっているのであろうか？出現頻度を調べてみよう．

```
10 DIM K(6)
20 FOR I=1 TO 100
30     U=INT(RND*6)+1
40     K(U)=K(U)+1
50 NEXT I
60 FOR J=1 TO 6
70     LPRINT J,K(J),STRING$(K(J),"■")
80 NEXT J
```

```
 1        23     ■■■■■■■■■■■■■■■■■■■■■■■
 2        15     ■■■■■■■■■■■■■■■
 3        16     ■■■■■■■■■■■■■■■■
 4        16     ■■■■■■■■■■■■■■■■
 5        16     ■■■■■■■■■■■■■■■■
 6        14     ■■■■■■■■■■■■■■
```

一様分布に従う確率変数の和の分布は，理論的に求められるが，それを実験でためしてみるのも面白い．以下に示すのは，そのプログラム例と実行例である．

```
10 DIM K(12)
20 FOR I=1 TO 100
30     U1=INT(RND*6)+1
40     U2=INT(RND*6)+1
50     R=U1+U2
60     K(R)=K(R)+1
70 NEXT I
80 FOR J=1 TO 12
90     LPRINT J,K(J),STRING$(K(J),"■")
100 NEXT J
```

```
 1        0
 2        3      ■■■
 3        8      ■■■■■■■■
 4        11     ■■■■■■■■■■■
 5        8      ■■■■■■■■
 6        15     ■■■■■■■■■■■■■■■
 7        17     ■■■■■■■■■■■■■■■■■
 8        19     ■■■■■■■■■■■■■■■■■■■
 9        11     ■■■■■■■■■■■
 10       7      ■■■■■■■
 11       0
 12       1      ■
```

正規分布に従う乱数が必要になることもある．それには各種の方法があるが，本機の場合，$\sin x$ や $\cos x$ は割合に時間がかかり，それに対してRNDは割合に速いので「[0，1]上の一様乱数を12箇加えて6を引けば，ほぼ$N(0,1)$に従う乱数が得られる」という性質を用いるとよい．以下に示すのは，それをさらに変換して，与えられた平均値μと標準偏差σの正規乱数を作るようにしたプログラムの一例である．

```
10 INPUT "mu=";MU
20 INPUT "sigma=";SIGMA
30 FOR I=1 TO 10
40    GOSUB "NORMAL"
50    LPRINT X
60 NEXT I
70 END
80 REM --- NORMAL ---
90 LABEL "NORMAL"
100 W=0
110 FOR II=1 TO 12
120    W=W+RND
130 NEXT II
140 X=MU+SIGMA*(W-6)
150 RETURN
```

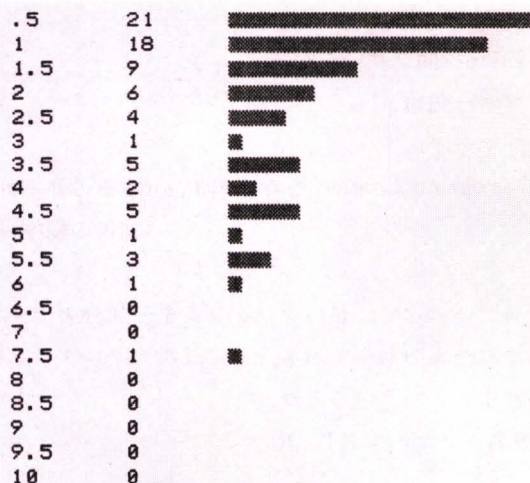
[mu ＝70, sigma ＝10 として実行した結果]

```
61.022691
66.236508
58.434823
67.175734
73.412837
52.094465
72.576536
69.350599
83.510256
82.491016
```

シミュレーションで指数乱数が必要になることもある．これは簡単でRNDの－L$\bar{\text{O}}$Gをとってパラメーターμを掛ければよい．

```
10 DIM K(20)
20 INPUT "mu=";MU
30 FOR I=1 TO 100
40     X=-LOG(RND)*MU
50     J=INT(2*X)
60     IF J<21 THEN K(J)=K(J)+1
70 NEXT I
80 FOR J=1 TO 20
90     LPRINT J/2,K(J),STRING$(K(J),"▓")
100 NEXT J
```

[mu ＝2として実行した例]

```
.5      21      ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
1       18      ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
1.5     9       ▓▓▓▓▓▓▓▓▓
2       6       ▓▓▓▓▓▓
2.5     4       ▓▓▓▓
3       1       ▓
3.5     5       ▓▓▓▓▓
4       2       ▓▓
4.5     5       ▓▓▓▓▓
5       1       ▓
5.5     3       ▓▓▓
6       1       ▓
6.5     0
7       0
7.5     1       ▓
8       0
8.5     0
9       0
9.5     0
10      0
```

RANDOMIZE文　　機種によっては，使用する度に毎回同じ乱数列が出て困ることがある．そういう場合に，強制的に乱数列を変更させる目的でRANDŌMIZE文というのがある．

本機の場合は，システムが出発値を毎回変えてくれるらしく，いつも違った乱数が出てくる．普通はその方がよいのであるが，場合によっては「前と同じ乱数列でもう一回やってみたい」ということがある．そういう場合，最初に

　　RANDŌMIZE　出発値

と書いておけば，指定された出発値を使ってくれる．出発値は－32768以上，65535以下でなければいけない．

付録Ⅰ　その他の命令，コマンド，関数

Ⅰ.1　命令およびコマンド

BEEP　ブザーを鳴らす

　　BEEP　　約1/3秒間ブザーを鳴らす

　　BEEP 1　ブザー開始

　　BEEP 0　ブザー停止

BOOT　IPL (initial program Loader)を呼び出す．（本書の範囲外）

CALL　機械語プログラムの呼び出し　（本書の範囲外）

CHAIN　続きのプログラムをディスクからロードして実行

　これはチェイン・ジョブといって，長いプログラムを一部分ずつメモリーに読み込んで実行するための命令である．単に「次のプログラムをロードして，すぐ実行せよ」というのであれば

　　LŌAD "装置名：ファイル名",R

あるいは

　　RUN "装置名：ファイル名"

という書き方もできるが，CHAINを使えば，データーの全部または一部を次のプログラムに渡すことができる，という利点がある．書き方は

　　CHAIN "装置名：ファイル名"

CLEAR　変数をすべてクリアする（CLRと同じ）

CLEAR アドレス　BASICが使用するメモリー領域を制限して機械語のプログラムの記憶場所を確保する　（本書の範囲外）

CLR　（CLEARと同じ）

CURSOR 左から何字目，右から何字目　（LŌCATEと同じ）

DEF CHR$　ユーザーが自分用のフォント（字母）を定義

DEF　CHR$（文字コード）＝文字列型変数名

の形で書く．フォントは，あらかじめ，次の要領で作って文字列型変数に入れておく．

- フォントは8×8ドット
- 文字列の第1字が最上列のドット・パターン
  第2字が次の列のドット・パターン
  …
  第8字が最下列のドット・パターン
  （点のある所が1，点のない所が0）
- じつは，上記が「青のドット・パターン」で，それに続く8字が「赤のドット・パターン」，その次の8字が「緑のドット・パターン」である（合計24字必要）．

DEF KEY　（KEYと同じ）

DEFUSR　機械語で書いた関数の開始番地を指定　（本書の範囲外）

ERROR　エラー・フラッグを立てる．

ŌN　ERRŌR　GŌTŌ　～　を用いるプログラムをデバックするとき，実際にエラー起こすのは大変だから，このERRŌR文によって，指定された番号のエラーが起きたのと同じ状態にする．火災訓練のときの発煙筒みたいな命令である．書き方は

ERRŌR　エラー・コード

FILES　ファイル一覧表を表示する

これはもともとディスクBASICのコマンドで，フロッピー・ディスクに記録されているファイルの名前の一覧者リスト・アップする機能をもつが，本機の場合，カセット・テープに使うこともできる．

カセット・テープをマウントし，巻き戻し，FILES ⏎ と指令すると，テープを読み，内容（プログラムやデーター）は早送りで読みとばしながらファイル名だけを読み込んで表示してくれる．なお，付帯情報として

ASCIIセーブされているか否か

書き込んだ日付と時刻

も表示してくれる.

KEY　ファンクション・キーの内容変更

KEY　番号，"文字列"

文字列の長さは15字までOK.

(例)「プリンターを3行分改行」を F3 にセットするには

```
KEY 3,"LP.:LP.:LP."+CHR$(13)
```

とすればよい.

KEY LIST　ファンクション・キーの内容一覧表を表示

本機の場合，その表示が KEY文の形のとおりになっているので，小さな変更だけなら，カーソルを修正箇所にもっていってなおし，⏎ を押せばよい.

KILL　ファイルの抹消

KILL　"装置名:ファイル名"

カセット・テープに対しては無効である.

KLIST　(KEY LISTに同じ)

LFILES　ファイルの一覧表をプリンターに出力.

機能についてはFILESの項を参照のこと（カセット・テープにも使える).

LINPUT　(LINE INPUTに同じ)

LOADM　機械語プログラムのロード　(本書の範囲外)

LOAD?　カセット・テープに正しくセーブされたか否かを検査する

プログラムをセーブした後，

```
APSS -1
```

で，今記録した先頭位置にもどして，

```
LOAD? ⏎
```

とすればよい．誤りの箇所があれば

```
Tape read error
```

と表示される.

MEM$　メイン・メモリーの指定された番地に文字列を書き込む.

MEM$ (先頭番地 , 字数) = 文字列型変数名または式

MERGE　プログラムを読み込み, 既にメモリーに入っているプログラムに追加する.

LŌADコマンドで読み込むと以前のプログラムが消えてしまうが, **MERGE**で読み込めば以前のプログラムは消えないで, 新しいプログラムが付け加わる形になる. ただし行番号が重複した場合は, あとから入力した方が優先される. 書き方は

MERGE　"装置名 : ファイル名"

なお, あとから読み込むプログラムはASCIIセーブしたもの(**SAVE**コマンドの最後に **,A**を付けてセーブしたもの) でなければいけない.

MON　機械語モニターを起動　(本書の範囲外)

NEW ON　機械語のプログラム等で使用するため, BASICの使用領域を制限する.

CLEAR文はBASICで使用する領域の上限を指定するが, **NEWŌN**文は下限 (先頭番地) を指定する.

NEW　ŌN　番地

OPTION BASE　配列の添字の下限を0にするか1にするかを指定

ŌPTIŌN　BASE　0または1

[**備考**]　1) 指定しなければ下限は0になる.

2) 最初の**DIM**文より前に置くこと

3) たった1箇の違いだから, 1にしてもメモリーの節約にはほとんど効果はない. しかし, 添字0を使わない場合, 下限を1と宣言しておくと, 何かのまちがいで添字が0になったときにエラー・メッセージが出るから, デバッグの際には役に立つ.

OUT　I/O ポートに出力　　　　（本書の範囲外）

　　ŌUT　ポート番号，出力すべき値*

POKE　指定した番地への書き込み

　　PŌKE　番地，書き込むべき値*

というのが標準的な書き方で，その場合，一つのPŌKE文で1バイトしか書き込めないが，本機では

　　PŌKE　先頭番地，先頭の値*，次の値*，…，最後の値*

という形で，いくつものデーターを連続した番地に書き込むことができる（おかげで非常に使い易くなった）．なお，命令**MEM$**同様な目的に使用することができる．

POKE @　VRAM 上の，指定した番地に書き込む

　　PŌKE@　先頭番地，値*

VRAM というのはビデオ RAM の略称で，画面に表示する文字や図形を記憶しているメモリーで，メモリー・マップは概略，次のようになっている．

番地(16進法)	使用目的
2000～	文字表示の属性(色その他)
3000～	文字表示の文字コード
4000～7FFF	グラフィック表示の青のドット
8000～BFFF	グラフィック表示の赤のドット
CŌŌŌ～FFFF	グラフィック表示の緑のドット

[簡単な使用例]　実用的には全く意味のないプログラムであるが，とてもきれいだから，一度実行してみるとよい．（挿絵参照）

```
10 FOR I=&H2000 TO &H2400
15    K=I MOD 256
20    POKE@ I,K
30    L=PEEK@(I)
40    IF K=L THEN PRINT HEX$(I)
50 NEXT I
```

---

*　8ビット，すなわち0～255の整数値に限る．

SAVEM　機械語プログラムのセーブ　（本書の範囲外）

VERIFY　（LŌAD?に同じ）

WAIT　入力ポートが指定された状態になるまで待つ（本書の範囲外）

WRITE#　テープやディスクに書き込む

PRINT#文を用いると，余分な空白が入ることが多く，しかも項目間の区切りが明確でないので，再入力できるようにするためには文字定数の形で項目間のコンマや文字列を囲む引用符を補なう必要があるが，WRITE#文だとそれを自動的にやってくれる．すなわち，

余分な空白はすべて省く

項目の間にはコンマが挿入される

文字列の前後には2重引用符が付く

I.2 関　　数

CDBL (数値)　倍精度実数型に変換

CHARACTER$ (左から何字目，上から何字目)　指定された位置に表示されている文字

CINT (数値)　整数型に変換

CSNG (数値)　実数型に変換

CSRLIN　現在，カーソルが画面の何行目にあるかを調べる．

EOF (ファイル番号)　ファイルの終端で真，他では偽となる．

FRE (ダミー変数)　ユーザー領域の残り（未使用）バイト数

INP (ポート番号)　I/O ポートからの 1 バイトを読み込む．

KANJI$ (漢字コード)　漢字のドット・パターン（16×16ドット．漢字 ROM 必要)．

　　POSITION文で表示位置を指定

　　K$=KANJI$（漢字コード）で漢字 ROM から読み出す

　　PATTERN　16，K$で表示

とすれば漢字を表示できる．

LPOS (ダミー変数)　現在プリンターの左から何字目まで打ったか(正確にいえば，プリンターに出力するためのバッファーの何字目まで書き込まれているか）を表す．

(使用例）単語を一つずつ読み込み，その字数を調べ，行の右にその単語を印字できる余白があればプリントし，余白がなければ改行してからプリントする．

```
10 REM --- LPOSノ ショウレイ ---
20 LINE INPUT A$
30 IF A$="*" GOTO 100
40 IF A$="" THEN LPRINT
50 N=LEN(A$)
60 IF LPOS(1)+N>39 THEN LPRINT
70 LPRINT A$;
80 IF LPOS(1)<40 THEN LPRINT " ";
90 GOTO 20
100 LPRINT
110 END
```

MEM$ (先頭番地, 字数)　指定された番地から始まる, 指定された字数の文字列を取り出す

PEEK (番地)　その番地のメモリーの内容（整数値とみなす）

PEEK @ (番地)　その番地の VRAM の内容

文字表示用の { 属性データーは&H2000～&H27FF
文字データーは&H3000～&H37FF

が割当てられている.

（使用例）文字表示の画面コピー・プログラムを作ってみよう. 本機のHCŌPYコマンドはユーザーが作ったフォントをコピーできるようになっているため, 文字が大きく, 行間が詰まってしまう. 標準のフォントだけでよければ, 下記のような簡単なプログラムで画面コピーできる（ここに示すのは80字／行のためのプログラムである. 40字／行の場合は行20と行40のコンスタントを変更すればよい).

```
10 FOR I=0 TO 24
20  FOR J=0 TO 79
30   K=&H3000+80*I+J
40   C=PEEK@(K)
45   IF C<32 THEN LPRINT " "; ELSE LPRINT CHR$(C);
50  NEXT J
60  LPRINT
70 NEXT I
80 END
```

POINT (横座標, 縦座標)　グラフィック画面座標で指定された点の色番号（正確にいえばパレット番号）

POS (0)　カーソルの現在の水平位置（左端が 0 ）

SCRN$ (行番号, 桁番号, 字数)　画面上の, 指定位置に表示されている文字（あるいは指定位置から始まる文字列）

SPACE$ (箇数)　指定箇数の空白を表す.

STRPTR　文字列データーを格納する領域の先頭番地

USR ルーチン番号 (引数)　ユーザーが機械語で書いた関数の呼び出し.

VARPTR（変数名）　　変数に対応するメモリー（格納場所）の番地（先頭アドレス）

ただし文字列型変数の場合には，**VARPTR**の値に**STRPTR**の値を加えたものがアドレスになる．

付録J　他機種のBASICとの主な相違点

実数型が5バイト　他の大部分の機種では単精度実数型のデーター1箇を4バイト(32ビット)で表しているが，本機では5バイト(40ビット)で表している．わずか1バイトの違い，と思われるかもしれないが，じつは大変な違いで，4バイトだと仮数部（有効数字）の桁数が24ビットしかないので演算精度は約7桁すなわち電卓以下であり，かなり注意深く計算を進めないと丸め誤差のために思わぬ失敗をすることがあるが，5バイトだと演算精度は9桁になり，トラブルはかなり少なくなる．そのかわり記憶場所をとる．また，数値計算以外の目的（たとえば図形表示のビット・パターンの記憶など）に使うときは，1データーあたりのビット数が他機種と違うから，既存プログラムのコンバージョンの際に注意が必要である．

行番号は1から　行番号に0は使えない．他機種では行番号0を許すものも多く，「行番号0はいくつでも書ける」という機種もある．そういう機種に慣れた利用者は頭の切換えが必要である．

長い識別名　変数名の長さ制限はなく(実際には文の長さが255字までだから，むやみに長いものは書けないが)，最後の字まで識別してくれる．ファイル名も「15字まで」と，他機種よりかなり長い．

ラベルを使える　GŌTŌ文などの行先の指定にラベル（文字列）を使用できる．N_{88}-BASICでもラベルが使えるけれども書き方が異なり，本機のはカナ文字を使えるし，また形が文字列型データーと共通なので

```
GŌTŌ A$
```

とか

```
GŌSUB A$(I)
```

などという書き方ができて，はるかに強力である．

2進定数を書ける　　2進法表現の定数を&Bという形で書ける．

(例)　&B10110011

プログラムを修正しても値が消えない　　他機種では，デバッグのため実行を中断し，プログラムを修正すると，変数の値はすべて初期化されてしまい，計算を最初からやりなおさなければならないのが普通であるが，本機ではプログラムを少しぐらい修正してもデーターは消えない．

倍精度関数計算可能　　引数が倍精度実数型ならば組込み関数の値は倍精度で計算される．これは，当然といえば当然だが，倍精度で計算されない機種もあるので念のため．

標準 TAB 設定間隔が 10 桁　　PRINT文において出力項目をコンマで区切ると，前の項目を出力後，次のTAB位置までスキップしてから次の項目が出力される．このTAB位置が，他機ではたいてい14字か15字間隔で設定されているのに対し，本機では10桁間隔で設定されている．これには良い点と悪い点がある．良い点は1行にたくさんの項目を表示できることで，特に　WIDTH 40　で使っている場合，他機種だとたいてい1行に2項目しか表示できないのに対し，本機だと4項目表示できる．ところが，1項目が常に10桁以下で表示できるとは限らない．そのため，「コンマによる区切り」が「上下をきれいにそろえる」という役に立たない．

(例)

```
10 LPRINT "N","1/N","SQR(N)
20 FOR N=1 TO 10
30     LPRINT N,1/N,SQR(N)
40 NEXT
```

```
N          1/N        SQR(N)
 1          1          1
 2          .5         1.4142136
 3          .33333333            1.7320508
 4          .25        2
 5          .2         2.236068
 6          .16666667            2.4494897
 7          .14285714            2.6457513
 8          .125       2.8284271
 9          .11111111            3
 10         .1         3.1622777
```

これはNと$1/N$と$\sqrt{N}$の数表を印刷するプログラムである.他機種では，これできれいに上下がそろうが，本機だと出力例のようになる．きれいに出力するにはTAB指定（またはUSING）を用いる必要がある.

IF文のTHENを省略できる　　本機では，IF文を

　　IF　条件式　THEN　文

の形で使うとき，THENを省略して，すぐに文を書ける.

　　IF　条件式　文

(例)　IF　N<5　PRINT　X

これは確かに簡単で良いけれども，標準規格違反で，大部分の他機種には通用しないから，こういう習慣は付けない方が良いと思う（PC-1500のように，THENを書くとエラーになる機種もあるが).

ON文は機能豊富　　本機のŌN文は,

　　ŌN　～　GŌTŌ　～,～,…

　　ŌN　～　GŌSUB　～,～,…

という普通の書き方のほかに

　　ŌN　～　RETURN　～,～,…

　　ŌN　～　RESTŌRE　～,～,…

　　ŌN　～　RESUME　～,～,…

という書き方ができる.

RETURN	行番号	指定された行番号に帰る
RESTŌRE	行番号	指定した所からREAD
RESUME	行番号	指定された行番号に帰る

は，他の機種にも見られるが，いずれもBASICの拡張機能である.

タイマーが2系統　　実時間時計TIME$と別にストップ・ウォッチTIMEがあるので，たいへん便利になった．実時間時計は，テレビの制御等にも使われるが，カセット・テープにプログラムやデーターを記録するとき，システムが日付と時刻を付記してくれるので，そういう情報を活用するためにも，常に正確な時刻を入れておくとよい.

付録K　省略記法一覧表

本機ではコマンドや命令や関数を省略形で入力することができる.
(例)　PRINT A,B,C のかわりに P.A,B,C でよい.
ほとんど全部のコマンド, 命令, 関数に省略形が用意されているが, たとえばABSをAB.と書くような, あまり利用価値のないものも多いので, ここでは役に立ちそうなものだけを抄録した.

正式の綴り	省略記法	正式の綴り	省略記法
AUTŌ	A.	MID$	MI.
CIRCLE	CI.	NEXT	N.
CŌLŌR	CŌL.	PAUSE	PA.
CŌNSŌLE	CŌNS.	PLAY	PL.
DATA	DA.	PRINT	P.
GŌTŌ	G.	REPEAT	REP.
HCŌPY	H.	RETURN	RE.
HEX$	HE.	RIGHT$	RI.
INPUT	I.	RUN	R.
LABEL	LA.	SAVE	SA.
LEFT$	LEF.	STŌP	S.
LFILES	LF.	UNTIL	U.
LIST	L.	WHILE	W.
LLIST	LL.	WEND	WE.
LŌAD	LŌ.	WIDTH	WI.
LPRINT	LP.	WINDŌW	WIN.
MERGE	M.	WRITE	WR.

索　　引

文法用語索引

著者略歴

戸川隼人（とがわ はやと）

1935年　東京に生まれる
　早稲田中学，早大高等学院を経て
1958年　早稲田大学第一理工学部数学科卒，科学技術庁航空技術研究所研究員。初年度は東大に出向して真空管式計算機 TAC の開発に従事。以後，数値計算法，計算機科学（図形処理，数式処理，文献検索等），ロケットの軌道計算，自動計測処理システム，構造解析等の研究を行なう。
　1965年より計算第1研究室長として計算センタ運営にあたる。
1971年　京都産業大学理学部計算機科学科助教授（72年教授）。数値解析学およびシステム工学を担当
1976年　日本大学理工学部数学科教授
　数値解析および計算機械論を担当

主要著書

マトリクスの数値計算　微分方程式の数値計算　有限要素法へのガイド　電子計算機概論　マイコンによる有限要素解析　詳解演習数値計算　数値計算入門
FORTRANによる有限要素法入門　演習 FORTRAN とその応用　有限要素法概論　数値解析とシミュレーション
共役勾配法　基本 BASIC とその応用
各種パソコン用 拡張 BASIC とその応用

パソコンライブラリ＝12

パソコンテレビ X1 BASIC

昭和58年6月10日 © 　　初版発行

著者　戸川隼人　　発行者　森平勇三
　　　　　　　　　印刷者　石野昭夫
　　　　　　　　　製本者　関川　弘

発行所　株式会社　サイエンス社
〒101　東京都千代田区神田須田町2丁目4番地
安部徳ビル
〔営業〕 ☎ (03) 256-1091(代)　振替 東京7-2387
〔編集〕 ☎ (03) 256-1093(代)

印刷　博文社　　製本　関川製本所

《検印省略》

3341-80621-2819